Outlier Detection: Methods, Models, and Classification

AZZEDINE BOUKERCHE and LINING ZHENG, University of Ottawa, Canada
OMAR ALFANDI, Zayed University, UAE

Over the past decade, we have witnessed an enormous amount of research effort dedicated to the design of efficient outlier detection techniques while taking into consideration efficiency, accuracy, high-dimensional data, and distributed environments, among other factors. In this article, we present and examine these characteristics, current solutions, as well as open challenges and future research directions in identifying new outlier detection strategies. We propose a taxonomy of the recently designed outlier detection strategies while underlying their fundamental characteristics and properties. We also introduce several newly trending outlier detection methods designed for high-dimensional data, data streams, big data, and minimally labeled data. Last, we review their advantages and limitations and then discuss future and new challenging issues.

1 INTRODUCTION

In the earliest days, detecting outliers was motivated by data cleansing: removing outliers from the dataset so that parametric statistical models would be able to fit the training data more smoothly. Soon more attention has been turned toward outliers themselves as outliers often represent interesting and critical information, e.g., cyber-attacks in a network, mechanical faults caused by defective industrial equipment, and so on. Hence, plenty of research efforts have been devoted to developing high-performance outlier detection techniques, which have been applied to a variety of real-life scenarios:

- Intrusion detection systems [1–4]: detecting unusual and malicious activities in computer systems or network systems, based on collected data such as operating system calls and network traffic.

- Fraud detection: identifying credit card frauds [5], financial transaction frauds [6], insurance claim frauds[7], and so on.
- Medical anomaly diagnosis [8, 9]: discovering potential disease risks or abnormal patient conditions based on the data collected by medical equipment.
- Fault or defect detection in industrial equipment and products [10, 11].
- Anomaly detection in wireless sensor networks [12]: detecting unexpected anomalous behaviors of the devices caused by unreliable power sources, unstable network connectivity, and so on.
- Anomaly detection in urban traffic flow [13]: identifying unexpected and deviant flow values that could be caused by traffic congestions, traffic accidents, and so on.

A more comprehensive list of applications can be found in surveys by Chandola et al. [14] and Wang et al. [15].

Outlier detection is challenging. One important reason is the lack of labeled data due to the rarity of outlier instances. Thus many methods are inherently unsupervised. A typical kind of outlier detection task is as follows: Given a set of data instances, identifying those instances (could be a fixed number) that deviate significantly from the rest. However, it is hard to propose a universal mathematical measurement for deviation that suits all datasets and scenarios. Moreover, due to the unsupervised nature, there is a gap between statistically eccentric instances and the instances of interest to users in real life. Over the years, a plethora of research works have emerged in the literature. On the one hand, state-of-the-art techniques (e.g., subsampling and ensembling [16], density peak clustering [17], deep learning [18], etc.) are being adopted to develop general, accurate, and efficient outlier detectors. On the other hand, with the rapid advancement of technologies in various domains (e.g., computer hardware, electronic devices, Internet applications, medical equipment, financial infrastructure, etc.), data come in larger quantities and higher complexity. Thus outlier detection faces new challenges: identifying outliers in data with extremely high dimensionality, in unbounded large volumes of data streams, and in distributed data of large scales, and so on.

1.1 Outlier Definition and Categorization

The first definition of the outlier is likely attributable to Grubbs in 1969 [19], "an outlier is one that appears to deviate markedly from other members of the sample in which it occurs." Other apt definitions include "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data" [20], "a data point which is significantly different from other data points, or does not conform to the expected normal behaviour, or conforms well to a defined abnormal behaviour" [14], and so on. Outliers are defined typically based on the following assumptions [21]: (1) Outliers are different from the norm with respect to their features; (2) outliers are rare in a dataset compared to normal instances.

Outliers and anomalies are the two most commonly used terms in the context of outlier detection. In fact, they are often used interchangeably [14, 22]. However, based on our own experience and understanding, a few subtle differences exist. In general, anomalies suggest a different underlying generative mechanism. In contrast, outliers tend to emphasize statistical rarity and deviation, and whether they are generated by a different mechanism is not straightforwardly addressed. In some cases in statistics and machine learning, outliers refer to those data instances (sometimes erroneous data points) that make it harder to fit the desired model. In such cases, people either remove outliers or use robust models (e.g., robust deep autoencoders [18]) to minimize the effect of outliers. From another perspective, in supervised learning, anomalies are the preferable term because there is reliable guidance to model the abnormal generative mechanism. In contrast, due to the lack of reliable guidance for unsupervised learning, methods usually rely on the intrinsic

distribution or structure of the data, to measure the deviation from the norm, hoping the discovered outliers represent the anomalies of interest. In this survey, we use outliers and anomaly interchangeably.

There are different ways to categorize outliers. First, based on the number of data instances involved to comprise a deviant pattern, there are (1) point outliers and (2) collective outliers [14]. A point outlier is an individual data instance that deviates largely from the rest of the dataset. This is the simplest type of outlier to identify and is the major focus of the research on outlier detection [14]. Collective outliers are a collection of data instances that appear anomalous with respect to the rest of the entire dataset. However, each instance within the collection may not constitute an outlier individually. An example of collective outliers is a specific sequence of access actions encountered in intrusion detection. Based on the scope of comparison, point outliers can further be classified into (1) local outliers and (2) global outliers. The notion of local outliers was first introduced in Local Outlier Factor (LOF) [23]. The detection of local outliers relies on the characteristic differences (e.g., the difference in neighborhood density) between the outlier and its nearest neighbors, whereas Global outliers address the difference with the entire dataset.

Based on the type of input data, outliers can be categorized into (1) vector outliers and (2) graph outliers [24]. Vector outliers are mentioned with vectorlike multi-dimensional data, while graph outliers exist in graph data. A vectorlike data point has multiple attributes, each of which has either a numeric value or a categorical value. The outlier detection methods rely on a distance definition between two vectorlike data points (e.g., Euclidean distance and Cosine distance). Graph data consist of nodes and edges, which well represent the inter-dependencies among data objects. Outliers in graph data can be point outliers (e.g., node outliers and edge outliers) or collective outliers (e.g., sub-graph outliers) [24]. This survey focuses on methods that address the outlier detection problem in vectorlike multi-dimensional data only. Readers can refer to Reference [25] for a comprehensive survey of outlier detection in graph data.

1.2 Classification of Methods and Scope of Discussion

Based on the availability of the input data's labels, outlier detection methods can be categorized into three types: (1) supervised outlier detection, (2) semi-supervised outlier detection, and (3) unsupervised outlier detection. Supervised outlier detection relies on labeled training data to build predictive models. Supervised outlier detection can be viewed as a binary classification problem, usually with imbalanced training data: many more instances in the normal class than in the outlier class. Semi-supervised outlier detection either uses training data with only normal labels (e.g., One-class Random Forest [26]) or builds models with a majority of unlabeled data and a small amount of labeled data (e.g., Das et al. [27]). Unsupervised outlier detection uses unlabeled data to either build models for outlier score calculation (e.g., Isolation Forest [28]) or directly calculate outlier scores for the input data without building models (e.g., LOF [23]).

This literature survey mainly focuses on unsupervised outlier detection. Unsupervised outlier detection is the most extensively researched type of outlier detection technique. This is due to the fact that labeled data for outlier detection are generally hard to obtain or even undesirable when the intention is to discover anomalous patterns that were never encountered before. However, recently, quite a few studies have explored how to incorporate human feedback into unsupervised models, and they have exhibited very promising improvement over purely unsupervised techniques. Accordingly, we will cover some of the newest semi-supervised methods in the end as well.

A classification of the methods discussed in this survey is illustrated in Figure 1. At a higher level, we categorize the methods into fundamental approaches and advanced approaches based on the fact that advanced approaches are developed upon the fundamental ones, to address new challenges. These challenges include high-dimensional data ("curse of dimensionality"), unbounded and dynamic data streams, big data in a distributed setting and effective usage of very limited
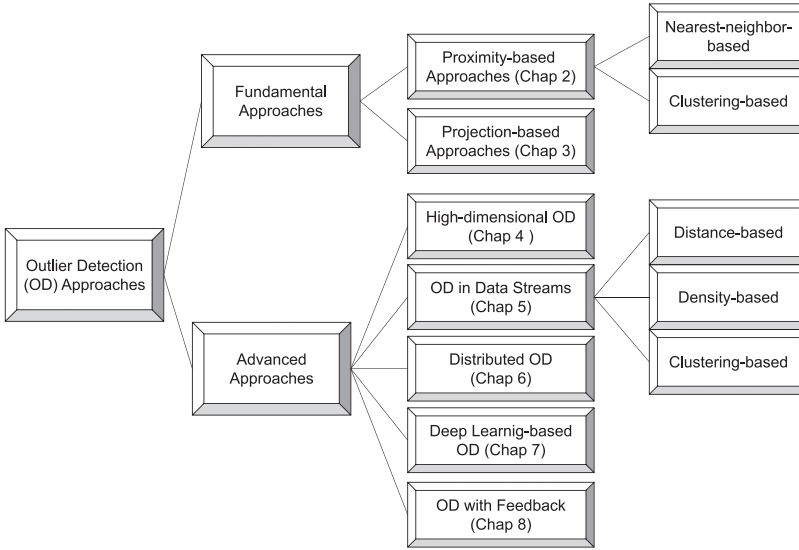
Fig. 1. Classification of outlier detection techniques.

labeled data. The fundamental approaches are further grouped into proximity-based methods and projection-based methods, according to the underlying techniques used by these methods. Proximity-based approaches rely on nearest-neighbor-based techniques or clustering algorithms to quantify an outlier's proximity to nearby data points, based on which the outlier score is defined. Projection-based methods adopt techniques such as LSH [29] and space-filling curves [30], to convert the original data into a new space/structure with reduced complexity, where the outlier scores are defined based on the characteristics of new space (e.g., number of data points in the same bin with LSH [29]).

Many studies (e.g., References [15, 22, 24]) in the literature adopt the terms "distance-based" and "density-based" for the categorization of the outlier detection methods. However, the meanings of the two terms are not consistent over these works. For instance, Reference [22] describes distance-based methods as those that define outlier scores based on the distances to the nearest neighbors. However, Reference [15] follows the definition by Knorr et al. [31]: Given a distance (radius) r and a real number p, an instance is deemed to be an outlier if there are less than p other instances within a radius r. This description of distance-based methods is similar to how Reference [22] describes density-based methods: Outlier scores are based on the number of data points in specific regions. An important reason for this inconsistency is that the notions of distance and density are closely related. For instance, LOF [23] is usually considered a density-based approach. However, the computation of the density in LOF is based on the distance to the nearest neighbors.

To avoid complexity and confusion, we unify the methods from both the "distance-based" and "density-based" categories and place them under the umbrella of "nearest-neighbor-based." This is because they all involve the notion of nearest neighbors. The neighborhood of a data point p is usually defined as k data points with the shortest distances to p or the data points within a region centered by p. However, in Section 5, we still use the legacy categories "distance-based" and "density-based" for outlier detection in data streams. This is because the distance-based approaches for data streams discussed in this article all follow the definition of outlier by Knorr et al. [31], which initiated the term "distance-based outliers." However, density-based approaches for data streams discussed in this article are all developed on top of LOF [23].

In the literature, there has been a considerable amount of effort on surveying outlier detection methods [14, 21, 22, 24, 32–35]. Some of them focus on a specific application scenario [33] or a specific type of method or data [34, 35]. For instance, Gogoi et al. [33] surveyed outlier detection methods applied to identify network anomalies. Gupta et al. [35] reviewed outlier detection methods for temporal data. Zimek et al. [34] focused on high-dimensional numerical data. There are also survey works that are very inclusive [14, 21, 22, 24]. However, they mainly discuss methods that were proposed more than five years ago. In recent years, many new methods and new trends have emerged as a result of the rapid advancement in big data techniques, distributed computing architecture, deep learning, and so on. It is worth the effort to review them. Our survey aims at providing a comprehensive and up-to-date overview of the outlier detection area. We review and compare both the classic and newly published methods in the conventional category: proximity-based. We put more focus on the new trends, such as projection-based methods, distributed methods, methods using a small amount of user feedback, and so on.

1.3 Paper Organization

The rest of the article is organized as follows. Proximity-based approaches are presented in Section 2. We then talk about projection-based approaches in Section 3. Section 4 introduces the techniques used for high-dimensional data. Section 5 discusses the outlier detection approaches in the scenario of streaming data. Section 6 focuses on the distributed approaches in the face of the big data challenge. Section 7 presents deep learning–based outlier detection approaches. Section 8 introduces the state-of-the-art semi-supervised approaches. Finally, Section 9 addresses some open challenges and concludes this article.

2 PROXIMITY-BASED APPROACHES

The proximity-based approaches identify outliers based on their relations with nearby data points. A common situation is that an outlier is located in a sparse area, with very few data points within a given distance, or the nearest data points are very far away. The notion of proximity can be defined in various ways. In this section, we focus on the techniques that address the proximity by nearest neighbors and clusters.

2.1 Nearest-neighbor-based Approaches

Nearest-neighbor-based outlier detection approaches measure the degree of abnormality on the basis of a data point's relation to its nearest neighbors. There are two main ways to define the neighborhood: k nearest neighbors (k-NN) and a neighborhood within a pre-specified radius, centered by a data point. The underlying assumption is that normal data instances are closer to their neighbors, thus forming a dense neighborhood, whereas outliers are far from their neighbors, thus sparsely populated.

In this section, we investigate several classical outlier detection approaches based on the nearest neighbors, as well as more recent approaches taking advantage of subsampling and ensembling. Table 1 is a summary of the nearest-neighbor-based approaches introduced in this section.

Some of the primitive nearest-neighbor-based approaches are very straightforward and intuitive. For instance, the approach by Ramaswamy et al. [41] uses the distance to the kth nearest neighbor as the outlier score. The method by Angiulli et al. [41] uses the sum of distances to the k-NN [42]. Knorr et al. [31] rely on the number of neighbors within a pre-defined radius of a data point. Because the degree of abnormality is compared in the context of the entire dataset, these methods detect global outliers. They assume that the density across different regions of the dataset is homogeneous. However, this assumption may not hold for many real-life datasets. Thus they

Table 1. Nearest-neighbor-based Outlier Detection

Algorithm	Features	Time Complexity	Local Outlier	Comments
LOF [23]	N/A	$O(N^2)$	Yes	First to address local outliers
COF [36]	Shortest path to connect neighbors	$O(N^2)$	Yes	Address non-spherical distributions
LOCI [37]	Count-based neighborhood density	$O(N^3)$	Yes	Free of parameters but high time complexity
INFLO [38]	Reversed nearest neighbors	$O(N^2)$	Yes	Improvement on border points between two areas with different densities
LoOP [39]	Assuming Gaussian distribution of distances to k-NN	$O(N^2)$	Yes	Interpretability of output
iNNE [40]	Subsampling and ensembling	$O(N\psi t)$	Yes	Highly efficient
LeSiNN [16]	Subsampling and ensembling	$O(N\psi t)$	No	Intuitive & highly efficient

[1] t denotes the number of sample sets.
[2] ψ denotes the number of instances within each sample set.

are often outperformed in terms of detection accuracy by approaches that take into consideration varied density [22]. The latter type of approach focuses on local outliers.

The LOF [23] is a well-known approach that first introduced the concept of local outliers and has inspired many subsequent works on local outliers. Local outliers are significantly different with regard to its closeby data points. The LOF score for a data instance is based on the average ratio of the instance's neighbor's density to that instance's density. In other words, the outlier score is the density normalized by the density of the neighbors. The normalization by the neighbors is how LOF addresses the local outlier. The detailed procedure of calculating the LOF score is described as below.

First, the k-NN need to be obtained for each data instance p. Second, the local reachability density (LRD) is calculated based on the average reachability distance from p to its k-NN:

$$LRD(p) = \left(\frac{\sum_{o \in N_k(p)} d_k(p,o)}{\mid N_k(p) \mid} \right)^{-1}, \tag{1}$$

where $N_k(p)$ is the k-nearest neighborhood of p and $d_k(p,o)$ is the reachability distance, which is defined as the larger value between the kth-nearest-neighbor distance to o (k-distance) and the distance between p and o, i.e.,

$$d_k(p,o) = \max \{k\text{-distance}(o), \text{distance}(p,o)\}. \tag{2}$$

The local reachability density is basically the reciprocal of the average distance to the neighbors unless there exist some neighbors that are "sufficiently close." The reason to introduce reachability distance other is to create a smoothing effect that reduces the statistical fluctuations of $d(p,o)$ for all the o's close to p [23]. Finally, the LOF score can be calculated by comparing the local reachability density (LRD) of p with all its k neighbors' LRDs:

$$LOF(p) = \frac{\sum_{o \in N_k(p)} \frac{LRD_k(o)}{LRD_k(p)}}{\mid N_k(p) \mid}, \tag{3}$$

which equals

$$LOF(p) = \frac{\sum_{o \in N_k(p)} LRD_k(o)}{\mid N_k(p) \mid LRD_k(p)}. \tag{4}$$

Informally, the LOF score of p is the average ratio of p's neighbors' density to p's density. Usually outliers have neighbors with a higher density. Thus outliers have LOF scores higher than normal one, and a higher score indicates an instance is more likely to be an outlier.

The Connectivity-based Outlier Factor (COF) [36] addresses the shortcomings of LOF, which assumes that the outlier pattern is only low density in a Euclidean distance-based spherical neighborhood. However, such a view of outliers is overly simplified, and outliers in other patterns of neighborhood relations may not be successfully identified. For example, the normal instances of a two-dimensional dataset distribute roughly along a straight line. An outlier lies astray from the straight line but still has a considerable density. This type of outlier will have a similar LOF score to the normal data points. To overcome this shortcoming of LOF, COF uses the notion of "isolativity," which is the degree that a data point is connected with others. To quantify the "isolativity," COF uses the "chaining distance," which can be viewed as the shortest path connecting the k neighbors and the data instance. Then the COF for a data point is its chaining distance normalized by the average of the chaining distance of its k-NN.

Papadimitriou et al. [37] proposed Local Correlation Integral (LOCI) based on the definition of local density, which is the count of neighbors within a radius r around a data point (r-neighborhood). They devised a related measure called Multi-Granularity Deviation Factor (MDEF). The MDEF with a given radius r for a data point p equals one minus the ratio of the local density of p to the average local density of the points in the r-neighborhood of p. MDEF represents the degree that the data point deviates from its neighbors in terms of local density. Another related measure is δ_{MDEF}, which is the standard deviation of the local density of the points in the r-neighborhood normalized by the average local density in the r-neighborhood. To determine whether a data instance is an outlier, with the radius r increasing in each iteration, the MDEF and δ_{MDEF} of the data point are calculated, and if MDEF is larger than three times of δ_{MDEF} in any iteration, the data point is labeled as an outlier. An advantage of LOCI is that it does not require parameters, for instance, k in k-NN, which is a crucial and difficult choice. Instead, it expands the radius of the r-neighborhood and derives a binary outlier label on the basis of the standard deviation of the MDEF. Thus, another advantage of LOCI is that it is free of outlier cutoff threshold that must be specified by users in other approaches. However, due to the iteration for the radius expansion, the time complexity is $O(N^3)$. Aware of the high complexity of LOCI, the authors have proposed an approximate method named aLOCI [37]. aLOCI approximates the neighborhood using a space partitioning grid, resulting in practically linear performance.

Influenced Outlierness (INFLO) [38] uses a reverse nearest neighborhood set (k-RNN) combined with the k-NN to compute the outlier score. The k-RNN of a data point p is the set of other instances whose k-nearest neighborhood includes p. Thus, the size of a k-RNN set is not necessarily k. The rest of the computation is similar to LOF: The outlier score is derived by dividing the local density of p by the average density of p's neighborhood. The incentive of incorporating k-RNN for outlier analysis is to address the limitation of LOF that LOF fails to appropriately score the instances on the borders of clusters with significantly different densities. As depicted in Figure 2, data point p is on the board of a dense region (right) and a sparser region (left). Most of the members of p's k-NN would be from the dense region, resulting in a high LOF score because the neighbors from the dense region have higher density. However, p is not supposed to be deemed as anomalous considering the sparser region. However, if we take into account the k-RNN as INFLO describes, the extended neighborhood set would also contain many members from the sparser region. Thus, a more reasonable outlier score will be assigned to p, and p will not be viewed as an outlier.

Kriegel et al. [39] proposed the Local Outlier Probability (LoOP), which outputs a probability that indicates the likelihood of a data point being an outlier. LoOP attempts to tackle the dilemma other methods face: How to choose the suitable cut-off threshold for outlier scores to distinguish between
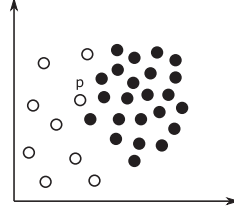
Fig. 2. INFLO addressing the limitation of LOF.

outliers and inliers. The formulated LoOP ranges from 0 to 1 with an interpretable meaning and thus can be more useful in practical scenarios. The computation framework of LoOP is similar to LOF: Compute local density and normalize it with neighborhood average. However, LoOP differs in the way it calculates the local density for a data point. It is assumed that a data point p is at the center of its neighborhood, and the distances to its k-NN follow a half-Gaussian distribution (distance is always non-negative). Accordingly, a quantity named standard distance is defined:

$$\sigma(p, N_k(p)) = \sqrt{\frac{\sum_{o \in N_k(p)} \text{dist}(p, o)^2}{|N_k(p)|}}, \tag{5}$$

where $N_k(p)$ is the k-NN of p. The standard distance resembles the deviation of $\text{dist}(p, o)$, where $o \in N_k(p)$. However, the standard distance uses 0 as the mean. Then the probabilistic set distance is used as the estimated density, which is defined as:

$$\text{pdist}(\lambda, p) = \lambda \cdot \sigma(p, N_k(p)), \tag{6}$$

where λ is merely a parameter controlling the contrast in the output scores without affecting the ranking. To normalize the density with regard to the average of the k-NN, Probabilistic Local Outlier Factor (PLOF) is defined as:

$$PLOF(p) = \frac{\text{pdist}(\lambda, p) \cdot |N_k(p)|}{\sum_{o \in N_k(p)} \text{pdist}(\lambda, o)} - 1. \tag{7}$$

Finally, to convert PLOF into a probability, normalization by deviation and a Gaussian error function is used. Ting et al. [43] pointed out in their work that nearest-neighbor-based outlier detection approaches are contrary to the conventional belief that more training data produce better results. Instead, using only samples from the original dataset gives rise to better performance for nearest-neighbor-based approaches. They argued that there exists an optimal sample size for an individual dataset. When the actual sample used is smaller than the optimal size, the data distribution is not well represented. But when the actual sample size increases above the optimal size, the resultant accuracy tends to decrease because the separation between normal data points and outliers diminishes. Put in another way, using small samples reduces the masking effect where outlier instances forming clusters are mistakenly regarded as normal instances [16, 28].

Based on subsampling, isolation using Nearest Neighbour Ensemble (iNNE) [40] creates isolation regions to determine outlier scores. An isolation model is built for each sample set. For each sample instance c within a sample set S, a hypersphere $B(c)$ is built with the sample instance at the center and the radius $r(c)$ as the distance between the sample instance and its nearest neighbor within the sample set. The isolation score for a data point p with regard to a sample S is defined as

$$I(p) = \begin{cases} 1 - \frac{r(nn(minS(x)))}{r(minS(x))} & x \in \bigcup_{c \in S} B(c) \\ 1 & \text{otherwise} \end{cases}, \tag{8}$$

where $minS(x)$ is the sample instance with the minimal hypersphere that x falls in, and $nn(c)$ is the nearest neighbor of c in the sample set. According to the equation, if the data point falls within the isolation hypersphere of any sample instance, the isolation score will be less than 1. The sample instance with the smallest radius is picked as the proxy of x. The score is then calculated as the radius ratio between the sample instance and the sample instance's nearest neighbor in the sample set. The comparative ratio of the radiuses is to address local outliers.

LeSiNN [16] is another outlier detection method that also builds models with subsampling. The outlier score for a data point p with regard to a sample set S is simply defined as the distance between p and p's nearest neighbor in S. Note that both iNNE and LeSiNN have a linear time complexity because the k-NN search for a data point is limited within a sample set, and the sample size is constant. Besides, both iNNE and LeSiNN use an ensemble to ensure the stability of the outlier detector. The final outlier score with the ensemble is the average score over multiple sets of samples.

Nearest-neighbor-based methods have the advantage of a more refined granularity on the outlier analysis over clustering-based approaches. This enables nearest-neighbor-based methods to differentiate between strong outliers and weak outliers that are more likely to be considered as noise [22]. However, high computation complexity usually comes as a cost, due to the expensive computation of the pairwise distances. Moreover, the choice of k has a significant impact on the performance. But the optimal choice of k varies for different approaches and datasets. An overly large k results in a weak distinction between outliers and normal points. An overly small k results in an unreliable estimation of the proximity density.

Using subsampling is a good way to reduce the time complexity to linear. Subsampling also helps with the aforementioned masking effect. Coupled with ensembling, subsampling-based methods can also deliver promising and reliable performance. However, the new problem is to decide the suitable sample size and ensemble size. Typically, when dealing with large datasets, a large ensemble size is desired for good performance. This could, however, cause a considerable increase in execution time.

2.2 Clustering-based Approaches

Clustering is an extensively studied data mining technique that groups data into multiple clusters with similar data instances ending up in the same cluster. Outlier detection algorithms based on clustering usually take a two-step procedure: grouping the data with clustering algorithms and analyze the degree of deviation based on the clustering results. As pointed as by Aggarwal [22], there is a complementary relationship between clusters and outliers, which can be simplistically put as that a data point not belonging to any clusters is considered an outlier. Aside from the cluster membership (whether or not in a cluster), there are two other commonly used cluster-related quantities to construct an outlier score. The first is the distance to the cluster center, the assumption being that normal data points are close to the cluster centers, whereas the outliers are far from them. The second is the cardinality of a cluster, the assumption being that the cluster of normal data points is dense and large, whereas the cluster of outliers is sparse and small.

Compared with nearest-neighbor-based approaches, a major advantage of clustering-based outlier detection is its efficiency. For instance, the time complexity for k-means clustering is $O(Nkt)$, with N data instances, k cluster centers and t iterations. Usually k and t are far smaller than n. In contrast, nearest-neighbor-based approaches typically induce quadratic time complexity due to the pair-wise distance computations. However, nearest-neighbor-based approaches depending on the point-to-point distance provide more refined granularity compared to clustering-based approaches, which employ simplified representations for aggregation of data points, e.g., cluster

Table 2. Clustering-based Outlier Detection

Algorithm	Clustering Algorithm	Features	Score Based on	Predefined Cluster #	Comments
Jiang et al. [44]	k-means [45]	Minimal spanning tree of cluster centers	cluster cadinality	Yes	Time efficient but coarse granularity & spherical clusters
CBLOF [46]	Arbitrary	Heuristic small and large clusters	Cluster cadinality & distance to cluster center	N/A	Too many parameters & misuse of cluster cardinality
LDCOF [47]	Arbitrary	Local normalization	Cluster cadinality & distance to cluster center	N/A	Detection of local outliers but too many parameters
Du et al. [17]	Density peak clustering [48]	Chebyshev's theorem for statistical threshold determination	Standard deviations of δ	No	Intuitive & arbitrary-shaped clusters

centers. In this section, we introduce some representative outlier analysis approaches based on clustering. They are summarized in Table 2.

Jiang et al. [44] presented an outlier detection approach based on a modified version of k-means clustering and a minimum spanning tree constructed from the cluster centers. The modified k-means clustering has an initial value and an upper bound for the number of clusters. If an encountered data point is far from all of the existing cluster centers, this data point will be assigned the center of a new cluster, which means the number of clusters increases by one. To determine how far is enough for the creation of a new cluster, two distances are involved. The first one is the shortest distance between any two cluster centers, which is maintained and updated when there are changes to the clusters. The second one is the distance between the data point and its nearest cluster center. A new cluster will be created if the first distance is no less than the second distance. When the actual number of clusters exceeds the upper bound, two clusters whose centers have the shortest distance will be merged into a single cluster. Similarly to k-means, the modified version also iterates through the entire dataset for a number of times, with the goal of minimizing the sum of the data point distance to its cluster center. As for the outlier detection phase, a minimum spanning tree is first created with the cluster centers as the nodes and their distance between one another as the edge weight. Then longest edges are repeatedly removed until the number of subtrees becomes k. The data points in the subtrees with the smallest cardinality are regarded as outliers.

The Cluster-based Local Outlier Factor (CBLOF) [46] is a clustering-based outlier detection approach that distinguishes small and large clusters by a quantitative measure. Given a set of k clusters $\{C_1, C_2, \ldots, C_k\}$, sorted by the decreasing order of the cluster cardinality, and two numeric parameters α, β, a boundary cluster C_b has at least one of the following two conditions hold: (1) $\sum_{i=1}^{b} |C_i| \geq \alpha|D|$; (2) $|C_b|/|C_{b+1}| \geq \beta$. Accordingly, the clusters after C_b in the sorted sequence are defined as small clusters, whereas the rest are large clusters. The intuition behind the first condition is that outliers account for only a small portion of the entire dataset. The second condition is due to the consideration that clusters with a high possibility of being outliers should be significantly smaller in size. Then the outlier score for data point p is defined on the basis of small clusters and large clusters:

$$\text{CBLOF}(p) = \begin{cases} |C_i| \cdot \min(\text{dist}(p, C_j)) & C_i \text{ is a small cluster} \\ |C_i| \cdot \text{dist}(p, C_i) & C_i \text{ is a large cluster} \end{cases}, \tag{9}$$

where $p \in C_i$, and C_j is a large cluster that does not include p. The cluster cardinality used as the scaling factor is intended to make the algorithm able to detect local outliers. The assumption is

that a larger cardinality is associated with a lower density. However, this does not hold in most cases. On the contrary, a large cardinality is supposed to indicate normality.

Later in the work by Amer et al. [47], it is demonstrated that simply removing the cluster cardinality of CBLOF can produce better results, which is named the unweighted-CBLOF:

$$\text{unweighted-CBLOF}(p) = \begin{cases} \min(\text{dist}(p, C_j)) & C_i \text{ is a small cluster} \\ \text{dist}(p, C_i) & C_i \text{ is a large cluster} \end{cases}. \tag{10}$$

This modification also makes unweighted-CBLOF a global outlier detector since the outlierness is evaluated with regard to the whole dataset. To introduce the local density characteristic, the authors of Reference [47] proposed Local Density Cluster-Based Outlier Factor (LDCOF), which uses the average distance of the data points within a cluster to the cluster center to normalize the outlier score:

$$\text{LDCOF}(p) = \begin{cases} \frac{\min(\text{dist}(p, C_j))}{\text{avg-dist}(C_j)} & C_i \text{ is a small cluster} \\ \frac{\text{dist}(p, C_i)}{\text{avg-dist}(C_i)} & C_i \text{ is a large cluster} \end{cases}, \tag{11}$$

where $p \in C_i$ and C_j is a large cluster that does not include p. The average distance of cluster members to the cluster center is defined as:

$$\text{avg-dist}(C) = \frac{\sum_{i \in C} \text{dist}(i, C)}{|C|}. \tag{12}$$

Note that both CBLOF and LDCOF have the incorporated clustering algorithm independent of the framework. But as suggested by [47], algorithms with a fixed number of clusters such as k-means are advantageous in performance and an overestimated number of clusters is recommended due to the potential non-spherical distributions.

Du et al. [17] devised a local outlier detection approach building upon the density peak clustering algorithm [48], which is a simple but effective density-based approach that can detect clusters of arbitrary shapes. The density peak clustering relies on two assumptions: (1) cluster centers have higher local density than surrounding data points and (2) cluster centers have a comparatively large distance to other data points with higher local density. The first assumption represents the concentration effect of a cluster, whereas the second assumption differentiates a cluster center and a nearby member in the same cluster. Two quantities are designed according to the two assumptions. The local density ρ for a data point is defined as the number of neighbor data points within a cutoff radius. The δ for a data point is its minimum distance to another data point with higher local density. In the clustering process, data points with high δ and high ρ are first assigned cluster centers, then each of the remaining data points belongs to the same cluster where its nearest data point with higher local density is assigned. After the clustering phase, the outlier detection approach herein calculates the mean and the standard deviation of δ within each cluster. Moreover, Chebyshev's theorem [49] is used to decide the deviation threshold for outliers.

3 PROJECTION-BASED APPROACHES

Many popular outlier detection techniques mentioned previously require the pairwise distance computation for the data points or the search for k-NN, which often incurs quadratic time complexity and makes those techniques hard to scale to very large datasets. In this section, we present approaches that use various projection techniques (e.g., random projection [50], LSH [29], etc.) to convert the original data into a new space with reduced dimensionality or complexity, while still preserving the proximity information (e.g., pairwise Euclidean distance, nearest-neighbor relations, etc.) of the original dataset to some degree. Then the outlier detection can be performed in the projected space with much-improved execution time.

Table 3. Projection-based Outlier Detection

Algorithm	Projection Technique	Base Outlier Detector	Features	Scalability to High Dimensionality	Comments
PINN [51]	Random projection [50]	LOF [23]	Approximate k-NN	Good	Improved time efficiency & high accuracy
LSOD [52]	LSH [29, 53]	kth-NN distance	LSH-based ranking & pruning	Medium	Early detection of top outliers
Schubert et al. [54]	Space-filling curve [30]	LOF [23]	Approximate k-NN & ensemble	Poor	Near-linear complexity & easy distributed deployment
Loda [55]	Sparse random projection	Histogram-based outlier detector	One-dimensional histogram ensemble	Good	Linear complexity & high accuracy & handling missing values
Isolation Forest [28]	Binary tree	N/A	Ensemble & subsampling	Good	Linear complexity & high accuracy
Extended iForest [56]	Binary tree	N/A	Random hyperplane cuts	Good	Improved accuracy

Table 3 is a summary of the approaches introduced in this section. Many of them are extremely efficient and also applicable to high-dimensional data. It is noteworthy that subspace techniques are also a type of straightforward projection. They have been widely used to address the challenges with high-dimensional data. Related techniques will be discussed further in Section 4.

Projection-indexed Nearest-neighbours (PINN) [51] is based on a random projection scheme to reduce the data dimensionality and thus decrease the computation cost of determining the k-NN relations. The random projection scheme they adopted was developed by Achlioptas [50] and can approximately preserve the Euclidean distances for pairs of data points with high probability. Each of the randomly and independently generated entries from the projection matrix is defined as:

$$a_{ij} = \sqrt{s} \begin{cases} 1 & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{s} \\ -1 & \text{with probability } \frac{1}{2s} \end{cases}, \tag{13}$$

where s is a parameter creating the effect that the random projection samples approximately $\frac{1}{s}$ of the entire feature space for each resulting projected feature. The advantage of random projections over other dimension reduction techniques such as PCA [57] is its efficiency. The authors of PINN further proved that the employed random projection could also preserve the k-distance of a data point and subsequently the neighborhood. These properties provide justification for their k-NN search in the projected space. The k-NN search is the most time-consuming component for many k-NN-based outlier detection algorithms. With the dimensionality decreased, not only are fewer data involved in the computation, but also efficient indexing structures (e.g., References [58, 59]) can be used to reduce the time complexity of k-NN search from $O(N^2)$ to $O(N \log N)$. Those indexing structures are not applicable in the case of high-dimensional data. After the approximate k-NN relations are determined, the data points are mapped back to the original space where the rest of the computation for LOF is conducted. To enhance the quality of the result, they maintain more than k nearest neighbors in the projected space, which are truncated to k for the computation in the original space.

Locality Sensitive Outlier Detection (LSOD) [52] leverages locality-sensitive hashing (LSH) [29, 53] to create an initial ranking of outliers. Locality-sensitive hashing (LSH) was first proposed by Indyk et al. [53] for the approximate nearest-neighbors problem in the Euclidean space. The property of LSH functions is that they map similar data points to the same hash buckets with

higher probability compared to those data points that are less similar to each other. The LSH function adopted by LSOD was introduced by [29]:

$$h(v) = \left\lfloor \frac{a \cdot v + b}{w} \right\rfloor, \tag{14}$$

where v is a d-dimensional data point, a is a d-dimensional random vector, each entry of which is independently selected from a p-stable distribution [60], b is a random real number chosen from the uniform distribution bounded by $[0, w]$, and w is a real number parameter, called the "width" of the function. LSOD uses LSH to project the original data into one-dimensional hash values. These hash values are then segmented into multiple LSH buckets. Then LSOD generates the ranking of outlierness for a data point based merely on the number of points that are mapped into the same bucket. The assumption behind is that outliers tend to have less similar data points and thus end up in buckets with a small number of data points. To efficiently identify the top outliers, LSOD integrates a number of pruning strategies for distance-based outlier detection, including PPSN [41], ANNS [41], and PPSO [61]. Data points with a higher ranking are processed first, which results in high thresholds for these pruning strategies and thus greatly improves the efficiency. The final outlier score is the distance to the kth-NN.

Another outlier detection algorithm based on projection was proposed by Schubert et al. [54]. To tackle the approximate nearest-neighbor search problem, they employed an ensemble of space-filling curves [30]. A space-filling curve maps a multi-dimension space into a single-dimension space. It has been widely used to develop indexing schemes for multi-dimensional data [62] and to perform similarity search in multi-dimensional space [63], and so on. Based on the idea that diversity improves the accuracy of outlier ensembles, the proposed algorithm herein creates numerous space-filling curves by varying the characteristics of the space-filling curve, such as employing different curve families, using different sets of subspaces and shifting offsets. Then all the data points are projected to each of the created space-filling curves and the resulting one-dimensional values are sorted on each of the space-filling curves, respectively. Based on the sorted sequence, a sliding window with a user specified width is used to produce candidates for each data instance on each individual curve. Finally, the candidates for each data point are merged together, and the k nearest ones are kept as the result. The authors argue that the space-filling curve is more suitable for k-NN search than other techniques, e.g., LSH [29] and random projection [64], due to the space-filling curve's preservation of closeness instead of distance or regions of a fixed size. Besides, they provided a distributed framework to scale the algorithm, where worker nodes perform the space-filling curve projecting and send samples to the master node for distribution estimation. Also, note that the proposed approximate k-NN search scheme can be used to accelerate outlier detection that is based on k-NN and reverse k-NN in a general sense. They chose to instantiate it with LOF [23] in the experimentation, which relies on k-NN search to estimate proximity density.

Loda [55] employs a line of sparse random projections. Each of the projection maps data points to a one-dimensional space, based on which histograms are generated to estimate the probability for each data point. It is important to know that Loda follows the spirit of ensembling and demonstrates how multiple weak outlier detectors combined together into an ensemble can produce very good results. More specifically, each sparse random projection is performed by calculating the dot products of the data instances and a random vector of dimension $\sqrt{d}$, where d is the dimension of the input data space. This means only a randomly selected portion of the features are involved for each projection. The elements of the projection vector are independently and randomly drawn from $N(0, 1)$. The rationale comes from the Johnson-Lindenstrauss lemma [65], which shows that such projection approximately preserves the pairwise L^2 norm distance (Euclidean distance) in the projected space. The histogram approximates the probability density of the projected

one-dimensional data by discretizing the projected data into equal-width bins. The number of data points residing in a bin leads to the estimation of the probability of the bin. Sampling is often used to construct the histograms. The output of Loda for a data instance p is an average of the logarithm of the estimated probabilities on the projection vectors:

$$S(p) = -\frac{1}{k} \sum_{i=1}^{k} \log\left(f_i\left(p^T v_i\right)\right), \tag{15}$$

where f_i is the probability estimator of the ith histogram and v_i is the corresponding projection vector. Loda can also handle missing variables for a data instance by taking into account only the histograms whose projection vector has a zero item on the place of that missing variable. Loda is not only very efficient but is also able to deliver high accuracy, thanks to the ensemble

$$S(p) = -\frac{1}{k} \sum_{i=1}^{k} \log\left(f_i\left(p^T v_i\right)\right), \tag{16}$$

where f_i is the probability estimator of the ith histogram and v_i is the corresponding projection vector. Loda can also handle missing variables for a data instance by taking into account only the histograms whose projection vector has a zero item on the place of that missing variable. Loda is not only very efficient but is also able to deliver high accuracy, thanks to the ensemble.

At the end of this section, we introduce some tree-based approaches. In a broad sense, the construction of the tree models can also be viewed as a type of projection, where the original data points are mapped to specific tree nodes, and those tree nodes contain proximity information about the original data.

Liu et al. [28] developed Isolation Forest, which is an unsupervised tree ensemble that intuitively resembles the random forest for classification problems. The Isolation Forest consists of multiple Isolation Trees (iTrees), which can be viewed as the unsupervised counterpart of decision trees. An iTree model is generated with a given sample set by recursively choosing one random attribute and one random split value of the data on every tree node until the height limit is reached or the terminal leaf contains one distinct data instance. The intuition behind is that outliers have a higher chance of being isolated on an earlier stage than normal data instances. Therefore, outliers are expected to have a shorter height in the isolation trees. Based on this idea, the outlier score of point p is defined as

$$Score(p) = 2^{-\left(\frac{\overline{d}(p)}{Ed(p)}\right)}, \tag{17}$$

where $\overline{d}(p)$ is the average depth of p in all the iTrees, and $Ed(p)$ is the expected length of the tree path for p. The latter is estimated based on the average length of the unsuccessful searches in the binary search tree. Isolation Forest is supposed to be constructed with small subsamples from the dataset rather than the entire dataset. Subsampling increases the diversity for the tree ensemble, which is beneficial for the accuracy of the result. Subsampling also helps alleviate or avoid the swamping (mistakenly identifying normal instances as outliers) and the masking (closely clustered outliers making themselves hard to be detected) issues. Another benefit of subsampling is the gain in efficiency since only a small portion of data is processed to build the model. After all, without having to deal with the pairwise distances, Isolation Forest is extremely efficient, with linear time complexity. Moreover, Isolation Forest also exhibits high detection accuracy, over a variety of datasets.

Hariri et al. [56] proposed Extended Isolation Forest to address the drawbacks of Isolation Forest. They provided an in-depth discussion about the limitations of axis-parallel cuts used in the original Isolation Forest, as well as on why the random hyperplanes benefit the algorithm. Extended

Isolation Forest differs from Isolation Forest in that it uses randomly generated hyperplanes involving multiple attributes to split the data and construct binary tree nodes, instead of using only one feature for each split. For each split, to determine whether a d-dimensional data point p should go to the left subtree or the right, the following equation is used:

$$(p - b) \cdot a \leq 0, \tag{18}$$

where b is the random intercept, each entry of which is drawn from a uniform distributed, bounded by the range of the corresponding attribute values of the data points in the tree node; a is a random vector deciding the slope of splitting, with each entry drawn from a normal distribution. Imagine the two-dimensional case where the separation can be visualized by lines. The splitting lines for Isolation Forest are all parallel to the coordinate axes, whereas those for Extended Isolation Forest have different angles. This flexibility in the slope makes Extended Isolation Forest capture the distribution and shapes better than Isolation Forest. Consider a specific two-dimensional example with two dense clusters: one on the top left corner and the other on the bottom right corner. Dense cuts will be made over the clusters. Since the Isolation Forest uses cuts that are parallel to the axes, this could easily create "densely cut areas" on the top right corner and the bottom left corner, which are unwanted artifacts. These two artifact areas will make the algorithm mistakenly assign low outlier scores for outliers within them. In contrast, Extended Isolation Forest is less likely to create such artifacts due to the variety of splitting slopes for separating the data.

4 HIGH-DIMENSIONAL OUTLIER DETECTION

As summarized by Zimek et al. [34], the challenges for outlier detection in high-dimensional data are twofold: the efficiency aspect and the effectiveness aspect. The difficulty in achieving efficiency with high-dimensional data is mainly attributed to two reasons. First, the similarity search such as k-NN search becomes more expensive in terms of computation cost because of the increased dimensions. Second, some techniques used to accelerate the outlier detection such as sampling [66, 67], pruning [68], ranking strategies [38, 69], and efficient indexing structures (R-trees [58], X-trees [59], etc.) degrade significantly or even introduce almost no improvement with high-dimensional data.

For the effectiveness aspect, the concern is whether the outlier detection method can identify meaningful outliers. A frequently used term related to this problem is the "curse of dimensionality" [34, 70–72]. It refers to the dilemma that in the high-dimensional space, the detection of outliers based on deviation tends to be interfered by a phenomenon called "distance concentration": The distances for all pairs of data points tend to become almost uniform. Thus all the regions in the dataset become almost equally sparse, and the distinction between outliers and normal instances is hard to capture. This phenomenon is caused by the dilution effects of a large number of "normally noisy" irrelevant dimensions/attributes [22]. In other words, these irrelevant dimensions conceal the information that can be used to identify outliers. This section focuses on approaches that are designed to tackle one or both of the challenging aspects of outlier detection in high-dimensional data. A summary of these approaches is presented in Table 4.

To improve the efficiency of outlier detection for high-dimensional data, Ghoting et al. [73] proposed Recursive Binning and Re-projection (RBRP). RBRP is inspired by ORCA [68], a nested loop outlier detection approach whose outlier score is based on the distance to the kth nearest neighbor. To take advantage of the pruning scheme by ORCA, k approximate nearest neighbors need to be found. RBRP uses a recursive binning process to accelerate the search for such approximate k-NN. First, the data points are recursively partitioned into bins until the size of an individual bin is smaller than a pre-defined threshold. This recursive partitioning strategy resembles divisive hierarchical clustering. More specifically, for each recursion of the partitioning, k-means is adopted

Table 4. Outlier Detection for High-dimensional Data

| Algorithm	Features	Improve efficiency/ effectiveness	Subspace	Comments
RBRP [73]	Recursive partitioning & approximate k-NN	Efficiency	No	Fast search of approximate k-NN
PINN [51]	Random projection & dimension reduction	Efficiency	No	Improved efficiency but approximate results
ABOD [74]	Angle variance	Effectiveness	No	High time complexity
Kriegel et al. [75]	Axis-parallel hyperplane	Effectiveness	Yes	Intepretability of result & High accuracy
HiCS [76]	Statistical test & ensemble	Effectiveness	Yes	Improved accuracy & generalized pre-processing method
RS-Hash [77]	Randomized hashing & ensemble	Efficiency & Effectiveness	Yes	Linear complexity & high accuracy & intepretability of results
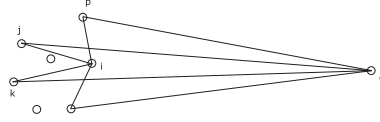
Fig. 3. The intuition of ABOD [74]

to create k partitions so that data points closer to each other in distance have a high probability of being assigned to the same bin. After the recursive partitioning phase, RBRP searches for k approximate nearest neighbors in the generated bins, where the data points are ordered as per their projection along the principle component to accelerate the search.

Note that the Projection-indexed Nearest-neighbours (PINN) [51] algorithm previously mentioned in Section 3 also aims at improving the efficiency of high-dimensional outlier detection. PINN leverages random projection for dimension reduction and uses approximate k-NN to deliver efficient performance.

Many more works in the literature focus on the effectiveness aspect of the high-dimensional outlier detection problem. Kriegel et al. [74] introduced an angle-based outlier detection method (ABOD) to address the issue of deteriorating quality encountered by Euclidean-distance-based algorithms in the face of high-dimensional datasets. As illustrated by Figure 3, the intuition behind ABOD is that if a data point is far from the rest of the data points (e.g., o), the angles having such a data point as the vertex (e.g., $\angle poq$ and $\angle jok$) tend to be small and vary slightly. In contrast, if a data point (e.g., i) is closely surrounded by others or is within a cluster, such angles (e.g., $\angle piq$ and $\angle jik$) usually have a high variance. Therefore, the outlier score for a data point relies on the variance of the angles having that data point as the vertex, weighted by the distances to the pair of other data points. The authors stress the importance of the distance weighting because naturally the angle to two data points varies more widely with a bigger distance. More specifically, the proposed angle-based outlier factor (ABOF) for data point i is defined as

$$ABOF(i) = VAR_{p,q\in D}\left(\frac{\overrightarrow{ip}\cdot\overrightarrow{iq}}{||\overrightarrow{ip}||^2||\overrightarrow{iq}||^2}\right)$$

$$= \frac{\sum_{p\in D}\sum_{q\in D}\frac{1}{||\overrightarrow{ip}||||\overrightarrow{iq}||}\left(\frac{\overrightarrow{ip}\cdot\overrightarrow{iq}}{||\overrightarrow{ip}||^2||\overrightarrow{iq}||^2}\right)^2}{\sum_{p\in D}\sum_{q\in D}\frac{1}{||\overrightarrow{ip}||||\overrightarrow{iq}||}} - \left(\frac{\sum_{p\in D}\sum_{q\in D}\frac{1}{||\overrightarrow{ip}||||\overrightarrow{iq}||}\frac{\overrightarrow{ip}\cdot\overrightarrow{iq}}{||\overrightarrow{ip}||^2||\overrightarrow{iq}||^2}}{\sum_{p\in D}\sum_{q\in D}\frac{1}{||\overrightarrow{ip}||||\overrightarrow{iq}||}}\right)^2,$$

where D is the dataset, the dot represents the dot product between two vectors. Since the outlier score for each data instance involves all the pairwise combinations of other data instances, this incurs the expensive $O(n^3)$ time complexity. To reduce the time complexity, two approximate variants were introduced: FastABOD and LB-ABOD. FastABOD constricts the selection of the pairs of data points for the outlier score computation to the k-NN of the data point. LB-ABOD is presented as a lower bound approximation of ABOD, which is designed to obtain the top outliers with the highest scores efficiently.

In addition, many works explore solutions on subspaces to handle the effect of the "curse of dimensionality," assuming that only a subset of the attributes is relevant for the discovery of meaningful outliers despite the rest of the attributes being noise. Kriegel et al. [75] developed an outlier detection schema that evaluates outlierness based on the deviation of an individual data point from the axis-parallel hyperplane spanned by a set of reference points. The hyperplane spanned by a set of points is associated with the subspace where these data points have high variance. The reference set for a data point is selected by ranking the number of shared nearest neighbors with the data point, based on the assumption that even though the traditional concept of k-NN loses its meaning in high-dimensional data, two data points generated by a similar mechanism still share a considerable number of nearest neighbors despite the irrelevant attributes. Note that the proposed method herein customizes the subspace for each data point because of the way how the hyperplane is created. Thus, the explanation for reasons of outlierness can be provided according to the related subspace.

Based on the assumption that outliers in high-dimensional data are hidden in multiple subspaces that exhibit non-uniformity and high contrast, Keller et al. [76] proposed a way of measuring the contrast of subspaces and accordingly a subspace search method called High Contrast Subspaces (HiCS). The contrast quantification for a candidate subspace is performed by sampling and statistical tests in an iterative way. In each iteration, a random attribute is chosen, and a random rectangular region in the subspace is generated. Then the deviation value is computed by comparing the marginal probability and the conditional probability. The final contrast value for a subspace is obtained by combining the individual deviation values. Based on the contrast quantification method, high-contrast subspace candidates are produced in a fashion that resembles the famous Apriori algorithm [78]. Starting from two dimensions, subspaces with a contrast value over a pre-defined threshold will be used for the candidate generation of the current dimension plus one. Then a pruning step that removes redundant subspaces ensues. As for the final outlier score, the results over different subspaces are combined, which resembles the feature bagging technique [79]. However, HiCS selects subspaces, whereas such selection in feature bagging is random. In the article, LOF is used as the actual outlier detector. However, the choice of the outlier detector is independent, and thus HiCS can be viewed as a generalized pre-processing technique for high-dimensional outlier detection.

Sathe et al. [77] proposed RS-Hash, an extremely efficient and accurate subspace outlier detection approach based on randomized hashing. RS-Hash follows the spirit of ensembling and averages the scores of all the ensemble component as the final score. Each component is essentially a set of models based on the closed hash function. Those models are trained by a sample of the original dataset, through a variety of randomized transformations and normalizations, coupled with randomized selections of the subspace. The outlier score for a data point output by an individual ensemble component is based on the number of sampled data points falling in the same hash bin during the training phase. Naturally, a low count in such bins indicates outlierness. Intuitively, RS-Hash estimates the density of the rectangular regions for a given data point, over different subspaces. Due to the randomization, the rectangular regions evaluated for a data point in different ensemble components are of different sizes, which is important to the accuracy of the

ensemble. Similar to the approach proposed by Kriegel et al. [75], RS-Hash also provides insights for the reason of a data point being an outlier, by analyzing the related subspaces that result in low scores. With linear time complexity, RS-Hash is considered a very efficient algorithm. Moreover, due to the use of subspace in the models, RS-Hash is also effective at handling the "curse of dimensionality."

Outlier detection for high-dimensional data is still a challenging problem due to the concerns about efficiency and effectiveness. Plenty of methods address either aspect or both with techniques such as approximate k-NN, subspace, and ensemble. Subspace-based approaches have recently received much attention in the research community. An inevitable problem to consider is how to identify the most meaningful and useful subspaces while minimizing the associated computational cost, given that the number of possible combinations of different attributes can be enormous.

In addition to the aforementioned methods, other interesting works in recent years include: HighDOD [80], using a dynamic subspace search method with a sample-based learning process; LODES [81], which relies on a novel local-density-based spectral embedding to identify outliers in nonlinear subspaces; RAMODO [82], which uses representation learning to reduce the dimensionality, and combines it with the random distance-based approach [16].

5   OUTLIER DETECTION IN DATA STREAMS

A data stream is a continuous and unbounded sequence of data in large volumes. Outlier detection in the context of data streams faces two major challenges. The first one is the storage memory challenge. As the data points continuously arrive and the sequence is theoretically endless, it is often not feasible to store the entire stream in the memory from the very beginning. Besides, the on-the-fly property of many outlier detection applications (e.g., intrusion detection in a computer network, suspicious behavior detection in wireless sensor network) imposes efficiency requirements.

To address these challenges, a commonly used technique is windowing: using a segment of the data stream, usually the newest one, to build incremental models and update the models in response to the change of involved data points. As summarized in Reference [83], there are four types of windowing techniques:

- Landmark window: A specific point in the data stream is fixed as the landmark. The outlier detection algorithm takes into account the sequence of data between the landmark and the current data point. Since the size of the data involved in the processing increases over time, memory storage becomes a major issue.
- Sliding window: A window of a fixed width w is sliding over the data stream. In other words, only the latest w data points are used as the context of outlier detection. Based on the definition of the window width, there are two types of sliding-window: count-based window and time-based window. The count-based window uses a fixed number of data points as the window width whereas the time-based window uses a fixed time duration.
- Damped window: A weight is assigned to each data point depending on the timing or order of its arrival. Usually, newer data points have higher weights so that the detection results can reflect the most recent trends.
- Adaptive window: Adaptive window is similar to the sliding window except that the window width w varies according to the rate of change from the data within the current window. The window expands when the data remain stationary and constricts when a change of the data is observed [84].

5.1   Distance-based Outlier Detection in Data Streams

A number of works in the literature have applied the distance-based outlier detection algorithm proposed by Knorr et al. [31] to the data stream scenarios. They adopt the same criterion for

Table 5. Distance-based Outlier Detection in Data Streams

Algorithm	Features	Memory Complexity	Outlier Storage	Count-based/ Time-based Sliding Window	Comments
STORM [85]	Safe inliers	$O(Wk)$	None	Count-based	High memory usage & high time complexity
Abstract-C [86]	Pre-handling neighbor counts	$O(W^2/S + W)$	None	Both	Imrpoved execution time
DUE [87]	Event queue	$O(Wk)$	Outlier list	Both	Efficient at re-evaluating outlier at runtime
MCOD [87]	Event queue & Micro-cluster	$O(cW + (1-c)kW)$	Outlier list	Both	Improved execution time & improved memory complexity
Thresh_LEAP [88]	Seperate index per slide & Minimal probing	$O(W^2/S)$	Outlier candidate list per slide	Both	Improved execution time but potentially more memory usuage

[1] k denotes the number of neighbors (as in k-NN).
[2] W denotes the size of the window.
[3] S denotes the slide size (the number of data points inserted and deleted for each time of sliding).
[4] c denotes the fraction of data points that are included in the micro-clusters.

determining outliers as in [31]: A data instance has less than p neighbors within a radius of r. This definition allows for unsupervised outlier detection without any assumptions on the distribution of the dataset. Table 5 summarizes the distance-based outlier detection approaches mentioned above.

Angiulli et al. [85] proposed a sliding-window-based approach called STORM to respond to outlier queries regarding the current window of data objects. STORM maintains a data structure called Indexed Stream Buffer to store the data instances in the current window and the related information of their neighbors. A neighbor of data instance i here is defined as another data instance whose distance to i is no bigger than the radius r. In Indexed Stream Buffer, each data point is associated with a list of neighbors that arrive before that data point, as well as the count of neighbors that arrive after that data point. An important property of this problem is identified that if a data instance has more than p neighbors succeeding it, it is guaranteed to be an inlier during its presence in the window. This property is used to develop two approximations in case of limited memory that is not capable of holding the entire window. The first approximation is to only keep a fraction of such guaranteed inliers in the window. The second approximation consists in reducing the size of the preceding neighbors for each data instance in the window. They conducted formal analysis on the approximation error bounds and proved the statistical guarantees of the approximations.

Yang et al. [86] developed another outlier detection approach named Abstract-C for data streams. They identified the challenge of pattern detection for data streams: The expiration of data points gives rise to complex changes in the existing patterns. More specifically, for the neighbor-based outlier detection, the problem becomes how to update the neighbor counts when data points expire due to the window sliding, without maintaining an exact neighbor list for each data point but only counts of neighbors. The provided solution takes advantage of the "predictability" of the neighbor counts for the succeeding windows. Abstract-C calculates the lifespan of each data point and preserves that information when updating the neighbor counts for that data point. Each data point is associated with its future neighbor counts for the next few window slides. In other words, the expiration of the data points is pre-handled, and no updating related to neighbor counts is required when they are eliminated from the sliding window. Abstract-C is more efficient and takes less space compared to STORM.

Kontaki et al. [87] introduced DUE and MCOD, event-based outlier detection approaches for data streams. DUE reacts to the expiration events to update the neighbor lists of only the related data points. DUE maintains a priority queue, named event queue, to store the unsafe inliers, which are those with less than k succeeding neighbors. The priority in the queue is determined by the earliest expiration time of the preceding neighbors. Also, an outlier list is maintained to keep track of all the current outliers. When the window slides, expired data instances trigger events to update the neighbor information of the unsafe inliers in the event queue. Some of them may become outliers and be moved to the outlier list. For newly added data points, a neighbor search is performed, and the resulting preceding and succeeding neighbor lists are created for each of them. Then they may be put into the outlier list or inlier list according to the number of their neighbors. The neighbors of the newly added data points also update their neighbor list, and they may be moved from the current queue depending on the neighbor counts. DUE is efficient at adjusting the outlier evaluation in response to the window sliding due to the event-based mechanism. However, the event queue takes extra time and memory to maintain the desired sorted order.

The specialty of MCOD [87] is its employment of micro-clusters to reduce the number of range query searches, i.e., searching for the neighbors of a data point within a radius. A micro-cluster is centered by one data point with a radius of $r/2$, comprising at least $k + 1$ member points. The data instances belonging to such micro-clusters are guaranteed to be inliers because every pair of data points has a distance of less than r, due to the triangular inequality of the distance metric. When the window slides, expired data points in the micro-clusters are removed. This can cause the dismission of a micro-cluster if it has less than $k + 1$ members after the removal of the expired data points. In such cases, the remaining data points are processed as newly added ones. New data points can be added to existing micro-clusters if they are within the distance. New points can also form a new micro-cluster after range query searches among the "free" points that are not included in any micro-clusters if there are at least k neighbors within the $r/2$-radius area of the new point. Aside from the points in micro-clusters, those "free" points are treated differently because they may have outliers and unsafe inliers. An event queue as in DUE [87] is maintained to manage the unsafe inliers. MCOD is advantageous in execution time thanks to the reduction of the neighbor searches. MCOD also needs less memory space since the points inside a micro-cluster do not need an extra neighbor list.

Cao et al. [88] proposed an approach called Thresh_LEAP to tackle the problem of distance-based outlier detection for data streams. Thresh_LEAP takes advantage of the temporal relationships among the individual slides and treats a slide as a unit for the neighbor counting. To be more specific, each data point maintains a list keeping track of the number of neighbors in each slide. Reciprocally, each slide has a trigger list storing the data points that will be affected when the slide expires. A strategy called "minimal probing principle" is adopted, through which the neighbors in the same slide are searched first, then the neighbors in the preceding slides are explored slide by slide from newest to oldest. The probing stops as soon as more than k neighbors are found. When a slide expires, the data points in the trigger are re-evaluated. If a data point has less than k neighbors due to the expiration, succeeding slides will be probed for the data point while preceding slides must have been probed already. The "minimal probing principle" as well as indexing each slide individually give Thresh_LEAP an advantage in CPU time, over other distance-based outlier detection methods except MCOD, which uses micro-clusters. However, considerable memory cost will be incurred, especially when small slide size creates a huge number of slides per window.

Even though the distance-based techniques are easy to understand and computationally efficient for data streams, they also have limitations. First, it is tricky to find the appropriate values of parameters r and p for different datasets. Also, it assumes homogeneous densities in the entire

Table 6. Density-based Outlier Detection in Data Streams

Algorithm	Features	Window Type	Time Complexity	Memory Complexity	Comments
Incremental LOF [89]	Selectively updating records	Landmark window	$O(N \log N)$	$O(Nk)$	High memory comlexity & high time complexity
MiLOF [90]	c-means-based summarization	Sliding window	$O(N \log W)$	$O(Wk)$	Low memory comlexity & time complexity but compromised accuracy
DILOF [91]	Density-based summarization	Sliding window	$O(NW)$	$O(Wk)$	Low memory comlexity & time complexity & high accuracy

[1] N denotes the size of the overall data stream.

[2] W denotes the width of the window.

dataset. However, for real-life datasets, approaches like LOF [23] that address local outliers may produce better results.

5.2 Density-based Outlier Detection in Data Streams

In this section, we introduce several outlier detection algorithms in data streams that are based on the density of the data points with regard to the k-NN. All of these approaches are extended from the LOF algorithm [23]. The distance-based approaches introduced in the previous section are considered to be able to detect global outliers since they assume homogeneous densities across the dataset. In contrast, as previously discussed in Section 2.1, LOF usually achieves good performance in datasets with non-homogeneous densities. This property also holds when it is used for data streams. Table 6 summarizes the density-based approaches for data streams introduced in this section.

Pokrajac et al. [89] proposed the first incremental version of LOF [23] for data streams. The Incremental LOF aims at delivering equivalent performance as applying the original LOF repeatedly on the data streams every time when a new data instance is received but with significantly less execution time. Inserting new data points and deleting obsolete data points (due to memory constraints or particular outdated behaviors) are followed by updating the records (k-distance, LRD, LOF score, etc.) of existing data points. Incremental LOF is based on an important observation that the insertion and deletion can only potentially affect a limited number of data points. To be more specific, the insertion or deletion of a data instance i influences the k-distances of the k-reverse-nearest-neighbors (k-RNN) of i. k-RNN of i is defined as the set of data points that have i as one of their k-NN. The change of the k-distances leads to the change of reachability distances and thus the LRDs of i's k-RNN's k-RNN, whose LOF scores need to be modified accordingly. They proved that the maximal number of k-RNN of a data point is proportional to k and exponentially proportional to the number of dimensions. Thus, if efficient approaches for k-NN and k-RNN queries (with time complexity $O(\log N)$) are applied, the overall time complexity of incremental LOF for the entire data stream of size N is merely $O(N \log N)$, if k and the dimensionality are treated as constants.

Salehi et al. [90] proposed MiLOF to overcome the unbounded memory issue of Incremental LOF by creating summarizations of previous data points, which leads to a fixed memory bound. MiLOF divides the available memory storage into two parts: one part for the newest b data points in the original form and the rest for c summaries of obsolete data points. Whenever the memory is running out, the older half of the b data instances are summarized and then removed from the memory. The summarization is performed using c-means clustering [45]. The cluster centers are chosen as the representatives for the clusters they belong to. These cluster centers also participate

in the LOF score calculation for the incoming data points as regular data points. However, the LOF-related records (k-distance, reachability distance, LRD, and LOF score) associated with these cluster centers are not computed based on their k-NN but based on the clusters they represent. To produce more accurate results, they introduced a flexible c-means that selectively summarizes the regions that are less likely to contain outliers, therefore, the regions with a higher probability of containing outliers are preserved in the original form. To fix the memory bound, the summaries in the form of cluster centers are merged in the same frequency of summarization so that only one single set of cluster centers exists. The merging is carried out with a clustering algorithm that weights the cluster centers according to the number of data points of the cluster. Subsequently, the updating of the LOF-related records for the merged cluster centers ensues. The insertion of incoming data points of MiLOF is similar to Incremental LOF but with modifications that handle the cases when the cluster centers appear in the k-NN of the incoming data points. MiLOF successfully reduces the memory consumption to a user-specified bound and decreases the time complexity accordingly. However, the accuracy is inevitably compromised due to the summarization, which may not effectively preserve the density of the data instances.

Another LOF-based outlier detection algorithm for data streams called DILOF was proposed by Na et al. [91]. DILOF also addresses the enormous memory consumption issue of Incremental LOF and additionally provides a solution for detecting a long sequence of outliers. Different from MiLOF, which uses c-means clustering for summarization, DILOF adopts a novel density-based sampling scheme to summarize the data, without prior assumptions on the data distribution. Thanks to this summarization technique, DILOF is shown to outperform MiLOF in detection accuracy. Similarly to MiLOF, DILOF has a detection phase and a summarization phase. In the summarization phase, a window of size W is maintained. As soon as the window is filled with data, $W/4$ out of the oldest $W/2$ data points are selected according to a proposed density-based sampling algorithm while the unselected points are removed to free up space of $W/4$. The goal of the summarization phase is to select the data points whose density resembles the original data to the highest possible degree. To achieve this end, they defined this task as a combinatorial optimization problem by extending a non-parametric Renyi divergence estimator [92] and converted the problem into a solvable binary constrained optimization problem. Then they introduced a new component for the objective function to preserve the data distribution. Furthermore, they developed a heuristic distance approximation technique, which was shown to greatly accelerate the summarization phase while still preserving the detection accuracy. As for the detection phase, they adopted the same method by Incremental LOF, which only updates the data points that are influenced when insertion or deletion happens. Moreover, they introduced a strategy called the skipping scheme to detect a group of outliers that comes in the form of long sequences. The skipping scheme shortcuts the detection process when a data point is found to be part of an outlier sequence. The assumption underlying the skipping scheme is that sequence outliers arrive consecutively, whereas the members of an emerging new class (considered to be normal data points) come in alternately with inliers.

MiLOF and DILOF have both managed to overcome the limitations of memory and execution time in Incremental LOF by summarizing a portion of the data points, which allows for keeping only a limited number of data points in the memory. Thanks to a better summarization technique, DILOF tends to outperform MiLOF in terms of accuracy measured by AUC. However, MiLOF seems to beat DILOF in time complexity. However, in the experiments in Reference [91], DILOF outperforms MiLOF in terms of execution time. This is because we treat the parameters related to c-means (the maximum number of iterations, number of cluster centers, etc.) as constants. In practice, when window width W is comparatively small to the c-means-related parameters, DILOF tends to outperform MiLOF.

Table 7. Clustering-based Outlier Detection in Data Streams

Algorithm	Features	Window Type	Score Based on	Predefined Cluster Number	Comments
D-Stream [93]	Grid-based clustering	Damped window	Density of a grid	No	Arbitrary-shaped clusters but poor scalability to high dimension
Elahi et al. [94]	k-means & delayed determination of outliers	Non-overlapping sliding window	Distance to cluster centers	Yes	Assuming spherical clusters & too many parameters
AnyOut [95]	Hierarchical clustering & ClusTree [96]	Damped window	Distance to cluster centers & Gaussian probability density	No	Real-time & varying granularity but assuming spherical clusters
Salehi et al. [97]	Hyperellipsoidal clustering	Non-overlapping sliding window	Whether belongs to a cluster	No	Time efficient & addressing switching data streams
Chenaghlou et al. [98]	Hyperellipsoidal clustering & Gaussian clusters	Non-overlapping sliding window	Gaussian probability density	No	Time efficient & addressing emerging distributions

5.3 Clustering-based Outlier Detection in Data Streams

As mentioned in Section 2.2, clustering-based approaches are advantageous over distance-based and density-based outlier detection in terms of time complexity. However, the granularity of outlier analysis is sacrificed. It is also noteworthy that the performance and the property of the outlier detection techniques depend heavily on the underlying clustering algorithms. For instance, a k-means-based approach may not be able to identify outliers in the context of arbitrary-shaped clusters. In the setting of data streams, new challenges include ensuring scalability, devising incremental strategies, and so on. In this section, we discuss outlier detection approaches for data streams that are based on clustering. We summarize the characteristics of these approaches in Table 7.

Chen et al. [93] introduced a grid-based approach called D-Stream for data stream clustering, which can also be used to detect and remove outliers. On the one hand, D-Stream maps the incoming data points to grids in an online fashion. On the other hand, D-Stream periodically forms clusters from the grids and eliminates the so-called "sporadic grids" that are considered to be outliers. Concretely, each existing data point is associated with a density coefficient, which decays with the elapse of time, to capture the dynamic changes of the data stream. The density of a grid is defined as the sum of the density coefficients of the data points residing in the grid. The grid density is updated when new data points are mapped to it. Due to the property of the density coefficient's decay factor, it is adequate to just maintain a characteristic vector for each grid rather than keeping track of all the density coefficients of the data points. To group the grids into clusters, the grids are classified into dense grids, sparse grids, and transitional grids according to the grid density. Based on the classification of the grids, a cluster is formed by connecting a group of dense grids that are surrounded by sparser grids. Due to the decaying factor, the class of a grid may change over time, which leads to the dynamic changes of the clusters. Therefore, the cluster structure needs to be periodically adjusted. With the assumption that outliers are mapped to grids with very few data points, they developed a threshold function to detect the grids with a density under a certain value. Note that the decaying factor can also result in low grid density even if the grid has a decent number of data points. This type of grid is not expected to be removed as outliers. Therefore, the threshold function is designed to distinguish this case from a grid having very few data points.

Elahi et al. [94] proposed an outlier detection algorithm based on k-means clustering for data streams. The data stream to be processed is treated as chunks of data, which is essentially the

non-overlapping sliding window model. There are two sets of cluster centers maintained throughout the stream processing. One set is called the actual cluster centers, which are the output of k-means clustering based on merely the current chunk of data. The other set is called the updated cluster centers, which are initiated as the average of previous updated cluster centers and current actual cluster centers, then run through k-means using both the current data chunk and candidate outliers. The candidate outliers are determined based on the distance of a data point to its updated cluster center. To address the scenario of pioneering members of an emerging cluster being falsely treated as outliers, the algorithm withholds the candidate outliers for L (user-specified parameter) chunks of data processing. The outlier scores for a candidate are accumulated during the course of L chunks, after which the candidate is finally judged to be an outlier or not. Therefore, only the cluster centers and candidate outliers are being held in memory while the data points considered to be safe are discarded. This strategy greatly reduces memory consumption. Even though this approach is intuitive and efficient, it has limitations such as requiring the determination of multiple parameters (k, L, outlier threshold, etc.) and assuming the spherical shape of clusters due to the use of k-means clustering.

Assent et al. [95] proposed an anytime outlier detection method called AnyOut, which leverages a tree structure built as the result of hierarchical clustering to represent the data and determine the outlier scores for the incoming data points in new window slides. AnyOut establishes a tree structure called ClusTree, which was developed by Kranen et al. [96] originally for parameter-free hierarchical clustering of data streams. Each tree node in ClusTree compactly represents a cluster by use of cluster features, which is a tuple composed of the number of the data points in the cluster, the linear sum of these points, and the squared sum of them. ClusTree is capable of updating the cluster features when new data points join the model. ClusTree also has additional buffer entries that allow for the anytime insertion of clusters. When it comes to building a ClusTree, they adopted a top-down bulk loading method [99], uses the expectation maximization [100] algorithm. AnyOut emphasizes the real-time characteristic of itself. With more time given, it outputs a finer-grained and more accurate score. This is achieved by a top-down outlier assessment strategy, in which the data point finds its closest cluster at each level. When the next data point arrives, and the result for the current data point must be returned, the outlier score is computed based on its relation with the most recently found cluster in the tree. Two ways to calculate the outlier score are provided. The first is called mean outlier score, defined as the distance between the data point and the mean of the entries in the cluster (resembling the centroids in k-means [45]). The second way is based on the Gaussian probability density of the data point.

Salehi et al. [97] introduced a weighted ensemble framework to detect outliers in data streams, which uses a clustering algorithm to model the normal patterns of data instances in previous windows. Their approach addresses the scenario where a data stream alternates between different states, each of which potentially consists of multiple distributions. The proposed framework comprises three components. First, all data points in the current window are clustered by the HyCARCE clustering algorithm [101]. HyCARCE is a density-based hyperellipsoidal clustering algorithm without predefined cluster numbers. HyCARCE outputs a set of cluster boundaries, which can be viewed as the built "clustering model." Every window is clustered, and their clustering models are kept in memory for the computation of the ensemble score for the incoming data points. The second component of the framework is to define the similarity between two clustering models, from which the ensemble weight is derived. To this end, the focal distance [102] between two hyperellipsoids is adopted. To be more specific, for two clustering models, the distance of every pair of hyperellipsoid boundaries, each from a different clustering model, is first computed. Then pairs of boundaries are selected out in a greedy fashion, starting from the shortest distance. In the end, the reciprocal of the sum of the resulting pairs' distances is used as the similarity between

two clustering models. The third component of the framework is to calculate the outlier score for a data point in the new window with the ensemble model, based on the relationship between the data point and previous clustering models. Specifically speaking, the algorithm checks if that data point belongs to any cluster for each clustering model in the history, based on whether the Mahalanobis distance between the data point and the cluster hyperellipsoid is beyond a threshold. The check produces a binary score for each previous clustering model. Then the final outlier score is the weighted sum of those binary scores, the weight being the similarity between the current clustering model and the corresponding former clustering model.

Another data stream outlier detection approach based on the HyCARCE clustering algorithm [101] was proposed by Chenaghlou et al. [98]. Different from the method presented in Reference [97], their approach models normal data patterns using Gaussian clusters and derives the outlier score based on the Gaussian probability density function of a cluster. Besides, the proposed approach is aware of and handles the newly emerging clusters. To process the data points in a new window, the first stage of the proposed approach is to find out if existing Gaussian clusters can explain the underlying distribution of some of the data points. To this end, they created two criteria: (1) The number of data points in the new window fitting the cluster must not be too few, which is tested by Cumulative Binomial Probability (CBP) function, and (2) the data points must spread out the cluster, which is tested by transforming the data points into standard Gaussian distributions [103], then into a spherical coordinate system [104]. The second stage is to detect potential emerging Gaussian clusters by using CBP, after removing the data points that can be explained by existing models in the first stage. If the result is positive, then HyCARCE clustering is employed to cluster these data points and save the new model. Finally, the score of a data point is the maximum value among the probabilities of the data point being observed under each of the Gaussian clusters.

6 DISTRIBUTED OUTLIER DETECTION

In the big data era, traditional centralized data mining and machine learning methods fall short for a few reasons. First, the resources of an individual computer may not be sufficient to perform the computation tasks due to limitations in disk storage, memory, CPU, and so on. Second, the centralized algorithms may not be able to satisfy the rigid time constraints required by many modern applications, e.g., real-time big data analytic applications. Moreover, the datasets themselves are tending to become more and more distributed.

In this section, we discuss recently proposed distributed-based outlier detection approaches that address the big data challenge. Table 8 summarizes the approaches discussed in this section. A challenging task for extending outlier detection to the distributed setting is minimizing the communication overhead while still guaranteeing the accuracy. This task is especially difficult for methods that require the computation of pairwise distances between data points. It is worth noting that in addition to the algorithms to be introduced subsequently, some works focusing on distributed k-NN search [105–109] can be inspirational for the development of distributed k-NN-based outlier detection algorithms.

Bhaduri et al. [110] developed DOoR, a distributed solution for the ORCA method [68], which uses the kth-nearest-neighbor distance as the outlier score and has a simple pruning rule. DOoR operates in a cluster of machines connected with a ring overlay topology, with an additional central machine that connects to all the machines in the ring. The central node maintains a list of current top-N points and the largest kth-nearest-neighbor distances as the cutoff threshold for pruning. Whenever the threshold information is updated in the central node, it will be broadcast to every machine in the ring topology for the most effective pruning. Each worker node in the ring contains a partition of the data. A worker node receives data partition from its previous node and validates

Table 8. Distributed Outlier Detection

Algorithm	Base Outlier Detector	Distributed Infrastructure	Features	Comments
Bhaduri et al. [110]	ORCA [68]	Network of ring topology	Top-N pruning	High communication overhead
Angiulli et al. [111]	Solvingset [112]	Ethernet network with TCP sockets	Top-N pruning	Impaired scalability due to broadcasting
DLOF [113]	LOF [23]	Hadoop MapReduce	Grid-based partitioning & duplication reduction	Reduced comminication overhead but not scalable to high dimensionality
DTOLF [114]	LOF [23]	Hadoop MapReduce	Grid-based partitioning & top-N pruning	Reduced comminication overhead but not scalable to high dimensionality
OW-OCRF [12]	One-class random forest [26]	Wireless sensor network	Weighted ensemble	Real-time response & high accuracy

the outlierness against its own local data partition, then passes it to the next node. After being passed around a circle, those data points not pruned will be sent to the central node for further evaluation. Essentially, all the data points not pruned are broadcast across the cluster, which incurs possibly high communication cost in the network and thus is not scalable to a very large scale of datasets.

Angiulli et al. [111] extended the SolvingSet algorithm [112] to the distributed environment. The solving set is an iteratively expanding sample set, based on which every data instance outside the set estimates their approximate k-NN in each iteration. A top-N outlier score is maintained and updated so that non-outliers can be pruned in advance. In the distributed setting, the solving set is broadcast across the cluster. The method also consists of the parallel computation and the synchronization of the partial results. This approach falls short in case of big datasets due to the correspondingly increasing size of the solving set to be broadcast.

Bai et al. [115] proposed a distributed solution for LOF. A grid-based partitioning method that tries to balance the workload and allocate the adjacent grids to the same machines is adopted. Based on the relation between the local k-distance and the grid borders, data instances whose k-NN all reside in the same partition can be identified. They are named "grid-local tuples." For other data instances, which are named "cross-grid tuples," they have devised a way to construct a minimum rectangle extension in every possible adjacent grid, which covers all the potential neighbors. Yan et al. [113] proposed a similar distributed version of LOF, named DLOF. DLOF greatly resembles the approach presented by Bai [115], but some extra optimizations are also introduced. DLOF uses grid-based data partitioning, with which each partition works in a fully distributed fashion. To ensure its compliance with the popular shared-nothing distributed infrastructures, the notion of supporting area is created so that every data instance p is distributed to other partitions where p is a k-nearest neighbor to some instances in those partitions. Moreover, some optimizations are introduced based on the observations that the necessary extending area for a data instance p cannot exceed the sphere with a radius of the local k-distance of p as well as that a data instance whose LOF score is computed and is not needed in any of the supporting areas can be eliminated.

The Distributed Top-N LOF (DTOLF) [114] provided a distributed solution for the Top-N LOF approach proposed in Reference [116]. DTOLF also utilizes grid-based data partitioning as DLOF [113]. It features a pruning strategy that eliminates data instances that are guaranteed not to be Top-N outliers and that are not needed by the computation of other machines. The pruning strategy takes into account the distances between the data points inside a partition and the

Table 9. Deep Learning-based Outlier Detection

Approach	Techniques	Data Label	Comments
Chen et al. [121]	Autoencoder & ensemble	Unlabeled	Increased computation cost
RDA [18]	Robust autoencoder	Unlabeled	High sensitivity to hyperparameter
AnoGAN [122]	GAN	Inlier label	high computation cost
ALAD [123]	GAN	Inlier label	Improved efficiency and accuracy
RAMODO [82]	Artificial neural network	Unlabeled	Customized representation but dependent on quality of candidate sets

boundaries of the partitions. Because the pruning strategy merely relies on the local data characteristics of a specific partition itself, it enables the reduction of communication cost among the machines in a cluster. Additionally, this elimination strategy reduces the data duplication rated compared to DLOF. To mitigate the problem of higher-dimensional data, they have introduced a correlation-aware partitioning, which is based on the observation that real-world datasets usually have strongly correlated dimensions. Thus, data partitioning can be carried out merely on independent dimensions.

A major limitation of the methods relying on grid-based data partitioning is that they do not scale well with the dimensions of the data. The number of grid cells grows exponentially with the increase of data dimensions. In the case of k-NN-based algorithms, each data instance may be needed by a great number of other grids or partitions to determine the k-NN neighborhoods. This usually incurs high communication cost across the cluster.

Moreover, Tsou et al. [12] presented a distributed unsupervised anomaly detection framework to address the challenges in anomalous behavior detection of wireless sensor network devices. Their approach relies on the one-class random forest [26], and the devices collaborate by sharing their models instead of data. To distinguish decision tree models according to their effectiveness, they developed an unsupervised ensembling algorithm to optimize the weights of the decision trees. The weights are learned by minimizing the uncertainty of the predictions for data points in an unsupervised fashion.

7 DEEP LEARNING-BASED OUTLIER DETECTION

Deep learning [117] is a class of techniques based on deep artificial neural networks. It is usually considered a subset of machine learning. Deep learning is a powerful tool to model nonlinear representations of data and has contributed to the advancement in a diversity of areas: speech recognition [118], natural language processing [119], object detection [120], and so on. Recently, there is a growing literature focusing on deep learning–based outlier detection methods. In this section, we introduce some of the most representative methods using deep learning (see Table 9).

An autoencoder is a type of artificial neural network used for unsupervised representation learning or nonlinear dimensionality reduction. The structure of an autoencoder is symmetric, usually with fewer nodes for the middle layers. The desired output of an autoencoder is the same as the input. In other words, the autoencoder is trained to reconstruct the input. Since the middle layers have fewer nodes, the autoencoder is forced to learn an efficient reduced encoding represented by the middle layers so that the input can be reconstructed as much as possible.

By assuming that outlier instances are more difficult to be reconstructed by the trained autoencoder, a straightforward way to leverage autoencoders for outlier detection is to use the reconstruction error as the outlier score [124]. However, since the training data can be contaminated by outliers in the unsupervised setting, the effectiveness of the model can be significantly weakened due to its sensitivity to outliers and the possible overfitting.

To address the autoencoder's susceptibility to overfitting when the training data contain outliers, Chen et al. [121] presented an ensemble-based outlier detection approach that combines the results of a group of autoencoders with diversity in the connectivity architecture. Because the variety of the base components is the key to the good performance of an ensemble, the autoencoders as the base models are designed to have the connections between the nodes randomly dropped. Moreover, these autoencoders are assigned different connection densities in a controlled way. Then the final outlier score is the median score among all the ensemble components. In this way, an individual autoencoder may encounter overfitting, but the result of the ensemble is much more robust. Additionally, an adaptive sampling strategy is adopted to accelerate the training and increase the diversity, which increases the sample size for each iteration as the training proceeds.

Robust Deep Autoencoder (RDA) [18] addresses the issue of contaminated training data by dividing the input data into two matrices, one containing outliers and the other being effectively reconstructed by the autoencoder. The idea is inspired by robust Robust Principal Component Analysis [125]. RDA not only provides high-quality representations from the autoencoder but also separates the outliers from normal instances. To achieve this separation, the autoencoder network is trained with a loss function that has two components. The first component is the reconstruction error for the normal matrix after the separation. The second component is based on the $l_{2,1}$ norm of the outlier matrix. The $l_{2,1}$ norm of a matrix is defined as

$$||M||_{2,1} = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} |M_{i,j}|^2 \right)^{1/2}. \tag{19}$$

The two components are balanced by a weight parameter λ, the choice of which is crucial to the performance of the method. As the loss function has two objectives, Alternating Direction Method of Multipliers [126] is used for the training. That is to alternatively optimize one objective while making the other fixed.

A typical/standard Generative Adversarial Net (GAN) [127] has two adversarial components, a generator neural network and a discriminator neural network. The generator is trained to learn a mapping from the latent space (e.g., sampled from a uniform distribution) to the data space. In other words, the generator generates fake data instances, which are used as negative training instances for the discriminator. However, the discriminator is trained to distinguish between the fake data generated by the generator and real data instances. GAN can be utilized for outlier detection because of its powerful ability to model complex data distributions.

Motivated to detect anomalies in image data as disease markers, Schlegl et al. [122] designed AnoGAN, which is a GAN that learns a representative model for normal data instances, accompanied by a scheme for anomaly scoring. The deep generative adversarial network is first trained on image data of healthy and normal examples. After the training, the generator learns a mapping from the latent space to the image space manifold, which represents the normal instances. To determine the degree of outlierness of an unseen input image, they developed a way to map the image space back to the latent space. More specifically, to calculate the position in the latent space for a new input image, at first, a random point from the latent space distribution is sampled. The random sample then is fed into the generator to generate an image. Based on a defined loss function, the gradients to update the coefficients of the point in the latent space are obtained. In this way, the point in the latent space can be optimized with backpropagation steps in an iterative manner. The loss function used for the mapping mentioned above is comprised of a residual loss and a discrimination loss. The residual loss is used to enforce the similarity between the generated image and the input image. The discrimination loss is based on the discriminator's intermediate feature representation. And the discrimination loss imposes the generator to generator data with

similar statistics to the training data. Finally, the anomaly score for an input image is the weighted sum of the residual loss and the discrimination loss at the final iteration of the previously mentioned mapping procedure. Intuitively, the anomaly score stands for how much the input image deviates from the model of normal instances.

Another GAN-based anomaly detection method is Adversarially Learned Anomaly Detection (ALAD) [123]. Similar to the idea of AnoGAN [122], ALAD also relies on GANs to model the distributions of normal instances, and the resulting anomaly score is based on the reconstruction errors, i.e., the dissimilarity between the original input data and their reconstructions. However, ALAD differs from AnoGAN in several aspects. First, ALAD is built upon bi-directional GANs [128, 129], which additionally learn an encoder network during the training. The encoder network is a mapping from the input data space to the latent space. Therefore, the latent representation for an input data instance can be obtained by simply passing the input data point through the encoder network, instead of having to solve an optimization problem using stochastic gradient descent for every input data instance as in AnoGAN. This is an important reason why ALAD outperforms AnoGAN in terms of computational efficiency. Second, ALAD also makes use of recent advancements to improve the performance of the bi-directional GAN, such as using an additional discriminator to encourage cycle-consistency [130] and applying spectral normalization to stabilize the training of the GAN [131]. Third, the reconstruction error-based anomaly score is different from AnoGAN. ALAD bases the anomaly score on the reconstruction error in the feature space of the cycle-consistency discriminator.

The deep learning–based methods presented so far use autoencoders or GANs to learn new representations of the data as the foundation for the subsequent outlier detection. The representations are yielded with a focus on retaining the normality information. However, there is a risk that the learned representations may not be relevant to the target outliers to be detected since there is no direct exposure or connection to the outliers. To address this problem, Pang et al. [82] proposed a deep neural network–based framework called RAMODO to learn representations for outlier detection guided by a loss function related to the target outlier detectors. The customization on the representation learning results in more effective representations and increases detection accuracy accordingly. More specifically, first, RAMODO uses a nearest-neighbor-based method (e.g., Sp [132]) to produce initial detection results. Then a thresholding technique (e.g., Cantelli's inequality [133]) is utilized to divide the data into normal candidates and outlier candidates. Second, meta triplet samples are generated from the normal candidate set and the outlier candidate set, which are fed into the input layer of the network. Finally, the representation learning is performed and optimized with the guidance of a carefully designed loss function that is based on the outlier score dissimilarity between the normal sample and the outlier sample on the learned representation. In other words, the outlier scores for both the normal sample and outlier sample from the meta triplet are computed with regard to the learned representation, using a target outlier detector. The loss function encourages the normal sample's outlier score to be smaller than the outlier sample's outlier score. A limitation of this proposed framework is its heavy dependence on the quality of the initial candidate sets. As pointed out by the authors, incorporating a few labeled data instances into the candidate sets would be very beneficial.

The main advantage of applying deep neural networks to outlier detection is the ability to extract representative features from complex and high-dimensional data, which delivers more accurate results compared to traditional approaches. However, typically a large data size is necessary for the deep neural networks to circumvent overfitting. But in many cases, data availability is limited. Moreover, many deep learning–based approaches are very sensitive to the hyperparameters. Tuning hyperparameters for optimal performances can be a challenging and time-consuming task.

Table 10. Outlier Detection with Feedback

Algorithm	Features	Base Outlier Detector	Active Learning Strategy	Comments
Görnitz et al. [136]	Tailored active learning strategy	SVDD [137]	Clusters near decision boundary	Robust against obfuscation & quick accuracy boost over a few labels
Das et al. [27]	AATP [138] & adjusted ensemble weights	Loda [55]	The most anomalous instances	High accuracy & generalizable
Vercruyssen et al. [11]	Label propagation	Constrained k-means based	Decision boundary	High accuracy
Siddiqui et al. [139]	Online convex optimization [140]	Isolation Forest [28]	The most anomalous instances	High accuracy & high efficiency & generalizable

8 OUTLIER DETECTION WITH FEEDBACK

The outlier detection methods mentioned so far do not need labeled data to train their models, i.e., they are unsupervised methods. Unsupervised outlier detection techniques have gained popularity both in the literature and in practice because typically labeled data are often scarce or unavailable. Moreover, sometimes such labeled data are even undesirable, because of the intention to discover anomalous patterns never encountered before. However, when there are labeled data available, even very few, the accuracy of the original unsupervised techniques can be significantly improved if the labeled data are properly utilized to adjust the existing models. This paradigm falls in the scope of semi-supervised learning [134]. A typical way to obtain and utilize the labeled instances is through active learning [135]: The initial model is built upon unlabeled data, based on which some data instances are selected by some query strategies to be labeled by a domain expert. Then the model is updated with the newly acquired label information. This type of feedback loop can be carried on iteratively until certain criteria are met. This section surveys some of the newest works in the literature that focus on incorporating human feedback into outlier detection to improve the detection accuracy. A summary of the approaches in this section can be found in Table 10.

In the work by Görnitz et al. [136], anomaly detection is regarded as an optimization problem named support vector data description (SVDD) [137]. SVDD computes a hypersphere to enclose the data, with radius r and center c. The hypersphere represents normality. The anomaly scores are based on the distances to the center c: data points found outside the hypersphere ball are considered outliers, whereas data points inside are viewed inliers. They present a generalized support vector data description making use of labeled data: Data points with inlier labels are required to reside within the hypersphere and vice versa. Thus it becomes a semi-supervised outlier detection problem. They show that the new optimization problem is unconvex but can be converted into a convex equivalent under mild assumptions. Additionally, different active learning strategies are introduced, which not only query the instances on the borderline but also those that could lead to the discovery of novel outlier categories.

Das et al. [27] proposed a semi-supervised approach that iteratively incorporates expert feedback into the model of an ensemble anomaly detection approach called Loda [55]. They aim at presenting the maximum number of anomalies to the expert. Thus, the instance with the highest anomaly score is selected for labeling in each iteration. The label information is then used to update the weights for the projections in Loda so that projections more effective at isolating anomalies are assigned higher weights. To achieve that effect, they devised an objective function modified from the accuracy at the top (AATP) approach [138]. The direct effect is that false positives are downgraded in the internal ranking based on the outlier scores produced by Loda, whereas true

positives are pushed up in the ranking. The proposed framework can be generalized for many other methods based on random projections besides Loda.

Vercruyssen et al. [11] described a semi-supervised anomaly detection approach that employs constrained k-means clustering [141] to perform the initial unsupervised scoring and iteratively updates the anomaly scores by incorporating expert labeling. In the clustering phase, the scoring formula is based on several intuitions: anomalies tend to deviate from its cluster centroid; the centroid of an anomalous cluster tends to deviate from other centroids; smaller clusters are more likely to bear anomalies. Then whenever new expert labels are available, the anomaly scores can be updated for unlabeled instances based on their distances to the labeled anomalies. This process is called label propagation. The underlying assumption is that unlabeled instances with shorter distances to the labeled anomalies should increase their scores compared to their peers. In label propagation, they introduce a weighting parameter to control the influence of the label information versus the score obtained from the clustering phase. To improve the detection accuracy, they used uncertainty sampling, which is choosing the unlabeled instances with a score closest to 0.5 for the expert to label.

Siddiqui et al. [139] proposed a general algorithm for anomaly detection that aims at aligning anomaly scores with the application-specific interestingness by incorporating expert feedback. They framed this anomaly detection problem with online convex optimization [140] and provided two loss functions that correspond to two different methods. The loss functions are associated with human expert feedback and promote the anomalies scores that are consistent with the feedback. A way to instantiate the algorithm with tree-based anomaly detection methods (e.g., isolation forest [28]) is described, which is achieved by adjusting the weights of the edges in the trees according to the feedback.

9 OPEN CHALLENGES

9.1 High-dimensional Outlier Detection

To perform outlier detection on high-dimensional data, one has to beware of the efficiency challenge as well as the infamous "curse of dimensionality" [70–72]. The efficiency challenge is mainly due to the enormous number of features involved in the computation as well as the infeasibility of applying efficient indexing structures such as R-trees [58]. A straightforward direction to address this challenge is using dimension-reduction techniques (e.g., PCA [57] and random projection [51]) that map the data into a lower-dimension space. How to design a dimension-reduction technique that effectively preserves the relevant information for subsequent outlier detection and how to tailor existing dimension-reduction methods for specific outlier detection algorithms or tasks are problems worth further investigation. As for the "curse of dimensionality," weakly relevant or irrelevant dimensions may create too much noise and thus conceal the outliers that could be identified with only relevant dimensions (subspaces), assuming such subspaces exist. Recent works making use of techniques such as subspace (e.g., Reference [76]), ensemble (e.g., Refeerence [77]) have demonstrated promising performance. However, the exploration of appropriate subspaces is still very challenging, as the number of possible combinations of attributes is exponential to the dimensionality.

9.2 Outlier Detection over Data Streams

Due to the infinite and dynamic characteristics, outlier detection over data streams faces many challenges. The most salient problem is how to scale despite the potentially infinite sequence of data. Some approaches choose to store the entire historical data in memory or secondary memory. However, this is apparently not feasible after some point. A solution is to remove the old

and obsolete data points and build models based on the most recent data. First, how to define the obsoleteness is a question. Moreover, how to ensure the accuracy if only a portion of the data is involved while the context should have been the entire history? Some methods tackle this problem by creating summaries to represent and retain important information about history data in a concise form, e.g., DILOF [91]. Another important issue is called concept drift, referring to the phenomenon that the distribution of the data points changes over time, leading to the change of the context for outliers. In such scenarios, an ideal outlier detection method is supposed to identify such changes to avoid alarming false positives.

9.3 Distributed Outlier Detection

As mentioned previously, distributed solutions are in demand because of the radically increasing data size. Thus many efforts have been made to extend centralized methods to the distributed setting. The biggest problem with such distributed solutions is the communication overhead, which is generally considered a vital factor, especially given very limited network bandwidth. This problem becomes even more acute with algorithms relying on pairwise distance, e.g., k-NN-based approaches because naively each data point will have to encounter every other for the distance computation, leading to a lot of broadcasting. To design efficient ways for conducting necessary data exchange between different machines, Yan et al. [113] turned to a grid-based partitioning scheme, which greatly improved the communication overhead but sacrificed the scalability to high dimensions. A possible solution would be approximate algorithms based on appropriate partitioning strategies. But this inevitably compromises accuracy.

10 CONCLUSION

In this article, we have presented a thorough classification of the past and recent outlier detection techniques and underline their fundamental characteristics and properties, and their advantages and limitations. Last but not least, we discuss future and open challenging issues related to outlier detection for complex and dynamic networks using AI-based technology.

REFERENCES

[1] Dit-Yan Yeung and Calvin Chow. 2002. Parzen-window network intrusion detectors. In *Object Recognition Supported by User Interaction for Service Robots*, Vol. 4. IEEE, 385–388.

[2] Robert Gwadera, Mikhail J. Atallah, and Wojciech Szpankowski. 2005. Reliable detection of episodes in event sequences. *Knowl. Inf. Syst.* 7, 4 (2005), 415–437.

[3] Mikhail Atallah, Wojciech Szpankowski, and Robert Gwadera. 2004. Detection of significant sets of episodes in event sequences. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM'04)*. IEEE, 3–10.

[4] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.* 28, 1 2 (2009), 18–28.

[5] Richard J. Bolton and David J. Hand. 2001. Unsupervised profiling methods for fraud detection. In *Proceedings of Credit Scoring and Credit Control VII*. 5–7.

[6] Sutapat Thiprungsri, Miklos A. Vasarhelyi, et al. 2011. Cluster analysis for anomaly detection in accounting data: An audit approach. *Int. J. Dig. Account. Res.* 11 (2011), 69–84.

[7] Clifton Phua, Damminda Alahakoon, and Vincent Lee. 2004. Minority report in fraud detection: Classification of skewed data. *ACM SIGKDD Explor. Newslett.* 6, 1 (2004), 50–59.

[8] Weng-Keen Wong, Andrew W. Moore, Gregory F. Cooper, and Michael M. Wagner. 2003. Bayesian network anomaly pattern detection for disease outbreaks. In *Proceedings of the 20th International Conference on Machine Learning*. 808–815.

[9] Jessica Lin, Eamonn Keogh, Ada Fu, and Helga Van Herle. 2005. Approximations to magic: Finding unusual medical time series. In *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*. Citeseer, 329–334.

[10] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. 2005. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery in Data Mining*. 401–410.

[11] Vincent Vercruyssen, Wannes Meert, Gust Verbruggen, Koen Maes, Ruben Bäumer, and Jesse Davis. 2018. Semi-supervised anomaly detection with an application to water analytics. In *Proceedings of the IEEE International Conference on Data Mining.*

[12] Yu-Lin Tsou, Hong-Min Chu, Cong Li, and Shao-Wen Yang. 2018. Robust distributed anomaly detection using optimal weighted one-class random forests. In *Proceedings of the 2018 IEEE International Conference on Data Mining.* 1272–1277.

[13] Youcef Djenouri, Asma Belhadi, Jerry Chun-Wei Lin, Djamel Djenouri, and Alberto Cano. 2019. A survey on urban traffic anomalies detection algorithms. *IEEE Access* 7 (2019), 12192–12205.

[14] Varun Chandola et al. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3 (2009), 15.

[15] Hongzhi Wang et al. 2019. Progress in outlier detection techniques: A survey. *IEEE Access* 7 (2019), 107964–108000.

[16] Guansong Pang, Kai Ming Ting, and David Albrecht. 2015. LeSiNN: Detecting anomalies by identifying least similar nearest neighbours. In *Proceedings of the 2015 IEEE International Conference on Data Mining Workshop (ICDMW'15).* IEEE, 623–630.

[17] Haizhou Du, Shengjie Zhao, Daqiang Zhang, and Jinsong Wu. 2016. Novel clustering-based approach for local outlier detection. In *Proceedings of the 2016 IEEE Conference on Computer Communications Workshops.* 802–811.

[18] Chong Zhou and Randy C. Paffenroth. 2017. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM International Conference on Knowledge Discovery and Data Mining.* 665–674.

[19] Frank E. Grubbs. 1969. Procedures for detecting outlying observations in samples. *Technometrics* 11, 1 (1969), 1–21.

[20] V. Barnett and T. Lewis. 1994. Outliers in Statistical Data (Probability & Mathematical Statistics). (1994).

[21] Markus Goldstein and Seiichi Uchida. 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS ONE* 11, 4 (2016), e0152173.

[22] Charu C. Aggarwal. 2015. Outlier analysis. In *Data Mining.* Springer, 237–263.

[23] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying density-based local outliers. In *ACM Sigmod Record*, Vol. 29. ACM, 93–104.

[24] Ji Zhang. 2013. Advancements of outlier detection: A survey. *ICST Trans. Scal. Inf. Syst.* 13, 1 (2013), 1–26.

[25] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: A survey. *Data Min. Knowl. Discov.* 29, 3 (2015), 626–688.

[26] Chesner Désir, Simon Bernard, Caroline Petitjean, and Laurent Heutte. 2013. One class random forests. *Pattern Recogn.* 46, 12 (2013), 3490–3506.

[27] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern, and Andrew Emmott. 2016. Incorporating expert feedback into active anomaly discovery. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining.* 853–858.

[28] Fei Liu et al. 2008. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining.* 413–422.

[29] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry.* ACM, 253–262.

[30] Guy M. Morton. 1966. A computer oriented geodetic data base and a new technique in file sequencing. *IBM Germany Scientific Symposium Series* (1966).

[31] Edwin M. Knorr and Raymond T. Ng. 1998. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, Vol. 98. Citeseer, 392–403.

[32] Victoria Hodge et al. 2004. A survey of outlier detection methodologies. *Artif. Intell. Rev.* 22, 2 (2004), 85–126.

[33] Prasanta Gogoi, D. K. Bhattacharyya, Bhogeswar Borah, and Jugal K. Kalita. 2011. A survey of outlier detection methods in network anomaly identification. *Comput. J.* 54, 4 (2011), 570–588.

[34] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. 2012. A survey on unsupervised outlier detection in high-dimensional numerical data. *Stat. Anal. Data Min.* 5, 5 (2012), 363–387.

[35] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. 2013. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.* 26, 9 (2013), 2250–2267.

[36] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W. Cheung. 2002. Enhancing effectiveness of outlier detections for low density patterns. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 535–548.

[37] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. 2003. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th International Conference on Data Engineering 2003.* IEEE, 315–326.

[38] Wen Jin, Anthony K. H. Tung, Jiawei Han, and Wei Wang. 2006. Ranking outliers using symmetric neighborhood relationship. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 577–593.

[39] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. 2009. LoOP: Local outlier probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management.* 1649–1652.

[40] Tharindu Bandaragoda. 2014. Efficient anomaly detection by isolation using nearest neighbour ensemble. In *Proceedings of the 2014 IEEE International Conference on Data Mining Workshop*. 698–705.

[41] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *ACM Sigmod Record*, Vol. 29. ACM, 427–438.

[42] Fabrizio Angiulli and Clara Pizzuti. 2002. Fast outlier detection in high dimensional spaces. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 15–27.

[43] Kai Ming Ting, Takashi Washio, Jonathan R. Wells, and Sunil Aryal. 2017. Defying the gravity of learning curve: A characteristic of nearest neighbour anomaly detectors. *Mach. Learn.* 106, 1 (2017), 55–91.

[44] Mon-Fong Jiang, Shian-Shyong Tseng, and Chih-Ming Su. 2001. Two-phase clustering process for outliers detection. *Pattern Recogn. Lett.* 22, 6–7 (2001), 691–700.

[45] John A. Hartigan and Manchek A. Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *J. Roy. Stat. Soc. Ser. C* 28, 1 (1979), 100–108.

[46] Zengyou He, Xiaofei Xu, and Shengchun Deng. 2003. Discovering cluster-based local outliers. *Pattern Recogn. Lett.* 24, 9–10 (2003), 1641–1650.

[47] Mennatallah Amer and Markus Goldstein. 2012. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proceedings of the 3rd RapidMiner Community Meeting and Conference (RCOMM'12)*. 1–12.

[48] Alex Rodriguez et al. 2014. Clustering by fast search and find of density peaks. *Science* 344, 6191 (2014), 1492–1496.

[49] Brett G. Amidan, Thomas A. Ferryman, and Scott K. Cooley. 2005. Data outlier detection using the Chebyshev theorem. In *Proceedings of the 2005 IEEE Aerospace Conference*. IEEE, 3814–3819.

[50] Dimitris Achlioptas. 2001. Database-friendly random projections. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, 274–281.

[51] Timothy De Vries, Sanjay Chawla, and Michael E. Houle. 2010. Finding local anomalies in very high dimensional space. In *Proceedings of the IEEE 10th International Conference on Data Mining (ICDM'10)*. IEEE, 128–137.

[52] Ye Wang, Srinivasan Parthasarathy, and Shirish Tatikonda. 2011. Locality sensitive outlier detection: A ranking driven approach. In *Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE'11)*. IEEE, 410–421.

[53] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*. ACM, 604–613.

[54] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. 2015. Fast and scalable outlier detection with approximate nearest neighbor ensembles. In *Proceedings of the International Conference on Database Systems for Advanced Applications*. Springer, 19–36.

[55] Tomáš Pevný. 2016. Loda: Lightweight on-line detector of anomalies. *Mach. Learn.* 102, 2 (2016), 275–304.

[56] S. Hariri, M. Carrasco Kind, and R. J. Brunner. 2018. Extended isolation forest. *ArXiv e-prints* (Nov. 2018). arxiv:1811.02141

[57] Gene H. Golub and Charles F. Van Loan. 2012. *Matrix Computations*. Vol. 3. JHU Press.

[58] Antonin Guttman. 1984. *R-trees: A Dynamic Index Structure for Spatial Searching*. Vol. 14. ACM.

[59] King-Ip Lin et al. 1994. The TV-tree: An index structure for high-dimensional data. *VLDB J.* 3, 4 (1994), 517–542.

[60] Vladimir M. Zolotarev. 1986. *One-dimensional Stable Distributions*. Vol. 65. American Mathematical Soc.

[61] Nguyen Hoang Vu and Vivekanand Gopalkrishnan. 2009. Efficient pruning schemes for distance-based outlier detection. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 160–175.

[62] Jack A. Orenstein and Tim H. Merrett. 1984. A class of data structures for associative searching. In *Proceedings of the 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*. ACM, 181–190.

[63] Ting Li et al. 2016. A locality-aware similar information searching scheme. *International J. Dig. Libr.* 17, 2 (2016), 79–93.

[64] Sampath Deegalla and Henrik Bostrom. 2006. Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In *Proceedings of the 5th IEEE International Conference on Machine Learning and Applications (ICMLA'06)*. IEEE, 245–250.

[65] William Johnson et al. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.* 26, 189-206 (1984), 1.

[66] George Kollios et al. 2003. Efficient biased sampling for approximate clustering and outlier detection in large data sets. *IEEE Trans. Knowl. Data Eng.* 15, 5 (2003), 1170–1187.

[67] Mingxi Wu and Christopher Jermaine. 2006. Outlier detection by sampling with accuracy guarantees. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 767–772.

[68] Stephen D. Bay and Mark Schwabacher. 2003. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of 9th ACM International Conference on Knowledge Discovery and Data Mining*. 29–38.

[69] Wen Jin, Anthony K. H. Tung, and Jiawei Han. 2001. Mining top-n local outliers in large databases. In *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 293–298.

[70] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When is "nearest neighbor" meaningful? In *Proceedings of the International Conference on Database Theory*. Springer, 217–235.

[71] Alexander Hinneburg, Charu C. Aggarwal, and Daniel A. Keim. 2000. What is the nearest neighbor in high dimensional spaces? In *Proceedings of the 26th International Conference on Very Large Databases*. 506–515.

[72] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. 2001. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the International Conference on Database Theory*. Springer, 420–434.

[73] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. 2008. Fast mining of distance-based outliers in high-dimensional datasets. *Data Min. Knowl. Discov.* 16, 3 (2008), 349–364.

[74] Hans-Peter Kriegel, Arthur Zimek, et al. 2008. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining*. 444–452.

[75] Hans-Peter Kriegel et al. 2009. Outlier detection in axis-parallel subspaces of high dimensional data. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 831–838.

[76] Fabian Keller, Emmanuel Muller, and Klemens Bohm. 2012. HiCS: High contrast subspaces for density-based outlier ranking. In *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE'12)*. IEEE, 1037–1048.

[77] Saket Sathe and Charu C. Aggarwal. 2016. Subspace outlier detection in linear time with randomized hashing. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'16)*. IEEE, 459–468.

[78] Rakesh Agrawal et al. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, Vol. 1215. 487–499.

[79] Aleksandar Lazarevic and Vipin Kumar. 2005. Feature bagging for outlier detection. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery in Data Mining*. 157–166.

[80] Ji Zhang and Hai Wang. 2006. Detecting outlying subspaces for high-dimensional data: The new task, algorithms, and performance. *Knowl. Inf. Syst.* 10, 3 (2006), 333–355.

[81] Saket Sathe and Charu Aggarwal. 2016. LODES: Local density meets spectral outlier detection. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 171–179.

[82] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. 2018. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining*. 2041–2050.

[83] Mahsa Salehi and Lida Rashidi. 2018. A survey on anomaly detection in evolving data: [with application to forest fire risk prediction]. *ACM SIGKDD Explor. Newslett.* 20, 1 (2018), 13–23.

[84] Albert Bifet and Ricard Gavalda. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 443–448.

[85] Fabrizio Angiulli and Fabio Fassetti. 2007. Detecting distance-based outliers in streams of data. In *Proceedings of the 16th ACM Conference on Conference on Information and Knowledge Management*. ACM, 811–820.

[86] Di Yang et al. 2009. Neighbor-based pattern detection for windows over streaming data. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. ACM, 529–540.

[87] Maria Kontaki et al. 2011. Continuous monitoring of distance-based outliers over data streams. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*. 135–146.

[88] Lei Cao et al. 2014. Scalable distance-based outlier detection over high-volume data streams. In *Proceedings of the 2014 IEEE 30th International Conference on Data Engineering*. 76–87.

[89] Dragoljub Pokrajac, Aleksandar Lazarevic, and Longin Jan Latecki. 2007. Incremental local outlier detection for data streams. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE, 504–515.

[90] Mahsa Salehi, Christopher Leckie, James C. Bezdek, Tharshan Vaithianathan, and Xuyun Zhang. 2016. Fast memory efficient local outlier detection in data streams. *IEEE Trans. Knowl. Data Eng.* 28, 12 (2016), 3246–3260.

[91] Gyoung S. Na, Donghyun Kim, and Hwanjo Yu. 2018. DILOF: Effective and memory efficient local outlier detection in data streams. In *Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining*. 1993–2002.

[92] Barnabás Póczos, Liang Xiong, and Jeff Schneider. 2012. Nonparametric divergence estimation with applications to machine learning on distributions. *arXiv preprint* (2012).

[93] Yixin Chen and Li Tu. 2007. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 133–142.

[94] Manzoor Elahi et al. 2008. Efficient clustering-based outlier detection algorithm for dynamic data stream. In *Proceedings of the 5th IEEE International Conference on Fuzzy Systems and Knowledge Discovery*. 298–304.

[95] Ira Assent, Philipp Kranen, Corinna Baldauf, and Thomas Seidl. 2012. Anyout: Anytime outlier detection on streaming data. In *Proceedings of the International Conference on Database Systems for Advanced Applications*. Springer, 228–242.

[96] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. 2009. Self-adaptive anytime stream clustering. In *Proceedings of the 2009 9th IEEE International Conference on Data Mining*. IEEE, 249–258.

[97] Mahsa Salehi, Christopher A. Leckie, Masud Moshtaghi, and Tharshan Vaithianathan. 2014. A relevance weighted ensemble model for anomaly detection in switching data streams. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining 2014*. 461–473.

[98] Milad Chenaghlou et al. 2017. An efficient method for anomaly detection in non-stationary data streams. In *Proceedings of the IEEE Global Communications Conference*. 1–6.

[99] Philipp Kranen and Thomas Seidl. 2009. Harnessing the strengths of anytime algorithms for constant data streams. *Data Min. Knowl. Discov.* 19, 2 (2009), 245–260.

[100] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc.: Ser. B* 39, 1 (1977), 1–22.

[101] Masud Moshtaghi, Sutharshan Rajasegarar, Christopher Leckie, and Shanika Karunasekera. 2011. An efficient hyperellipsoidal clustering algorithm for resource-constrained environments. *Pattern Recogn.* 44, 9 (2011), 2197–2209.

[102] Masud Moshtaghi et al. 2011. Clustering ellipses for anomaly detection. *Pattern Recogn.* 44, 1 (2011), 55–69.

[103] Richard Johnson et al. 2002. *Applied Multivariate Statistical Analysis*. Prentice–Hall, Upper Saddle River, NJ.

[104] David Henderson et al. 1994. *Experiencing Geometry on Plane and Sphere*. Tech. Report T-MATH, Cornell Univ.

[105] Wei Lu, Yanyan Shen, Su Chen, and Beng Chin Ooi. 2012. Efficient processing of k nearest neighbor joins using mapreduce. *Proceedings VLDB Endow.* 5, 10 (2012), 1016–1027.

[106] Chi Zhang, Feifei Li, and Jeffrey Jestes. 2012. Efficient parallel kNN joins for large data in MapReduce. In *Proceedings of the 15th International Conference on Extending Database Technology*. ACM, 38–49.

[107] Marius Muja and David G. Lowe. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 11 (2014), 2227–2240.

[108] Georgios Chatzimilioudis et al. 2016. Distributed in-memory processing of all k nearest neighbor queries. *IEEE Trans. on Knowledge and Data Engineering* 28, 4 (2016), 925–938.

[109] Caitlin Kuhlman, Yizhou Yan, Lei Cao, and Elke Rundensteiner. 2017. Pivot-based distributed k-nearest neighbor mining. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 843–860.

[110] Kanishka Bhaduri, Bryan L. Matthews, and Chris R. Giannella. 2011. Algorithms for speeding up distance-based outlier detection. In *Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining*. 859–867.

[111] Fabrizio Angiulli, Stefano Basta, Stefano Lodi, and Claudio Sartori. 2013. Distributed strategies for mining outliers in large data sets. *IEEE Trans. Knowl. Data Eng.* 25, 7 (2013), 1520–1532.

[112] Fabrizio Angiulli, Stefano Basta, and Clara Pizzuti. 2006. Distance-based detection and prediction of outliers. *IEEE Trans. Knowl. Data Eng.* 18, 2 (2006), 145–160.

[113] Yizhou Yan, Lei Cao, Caitlin Kulhman, and Elke Rundensteiner. 2017. Distributed local outlier detection in big data. In *Proceedings of the 23rd ACM International Conference on Knowledge Discovery and Data Mining*. 1225–1234.

[114] Yizhou Yan, Lei Cao, and Elke A. Rundensteiner. 2017. Distributed Top-N local outlier detection in big data. In *Proceedings of the IEEE International Conference on Big Data (Big Data'17)*. IEEE, 827–836.

[115] Mei Bai, Xite Wang, Junchang Xin, and Guoren Wang. 2016. An efficient algorithm for distributed density-based outlier detection on big data. *Neurocomputing* 181, C (2016), 19–28.

[116] Yizhou Yan, Lei Cao, and Elke A. Rundensteiner. 2017. Scalable Top-n local outlier detection. In *Proceedings of the 23rd ACM International Conference on Knowledge Discovery and Data Mining*. 1235–1244.

[117] Ian Goodfellow et al. 2016. *Deep Learning: Speech Recognition*. MIT Press.

[118] Dario Amodei et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of the International Conference on Machine Learning*. 173–182.

[119] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* 13, 3 (2018), 55–75.

[120] Tsung-Yi Lin et al. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2117–2125.

[121] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. 2017. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 90–98.

[122] Thomas Schlegl et al. 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Proceedings of the International Conference on Information Processing in Medical Imaging*. Springer, 146–157.

[123] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. 2018. Adversarially learned anomaly detection. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 727–736.

[124] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. 2002. Outlier detection using replicator neural networks. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery*. Springer, 170–180.

[125] Emmanuel J. Candès et al. 2011. Robust principal component analysis? *J. ACM* 58, 3 (2011), 11.

[126] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

[127] Ian Goodfellow et al. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. 2672–2680.

[128] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. 2016. Adversarially learned inference. *arXiv preprint arXiv:1606.00704* (2016).

[129] Donahue et al. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782* (2016).

[130] Chunyuan Li, Hao Liu, Changyou Chen, et al. 2017. Alice: Towards understanding adversarial learning for joint distribution matching. In *Advances in Neural Information Processing Systems*. 5495–5503.

[131] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* (2018).

[132] Mahito Sugiyama and Karsten Borgwardt. 2013. Rapid distance-based outlier detection via sampling. In *Advances in Neural Information Processing Systems*. 467–475.

[133] Devdatt P. Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press.

[134] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (Chapelle, O. et al., eds.; 2006) [book reviews]. *IEEE Trans. Neur. Netw.* 20, 3 (2009), 542–542.

[135] Burr Settles. 2009. *Active Learning Literature Survey*. Technical Report. T-CS, University of Wisconsin—Madison.

[136] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. 2013. Toward supervised anomaly detection. *J. Artif. Intell. Res.* 46, 1 (2013), 235–262.

[137] David M. J. Tax and Robert P. W. Duin. 2004. Support vector data description. *Machine Learning* 54, 1 (2004), 45–66.

[138] Stephen Boyd, Corinna Cortes, Mehryar Mohri, and Ana Radovanovic. 2012. Accuracy at the top. In *Advances in Neural Information Processing Systems*. 953–961.

[139] Md Amran Siddiqui et al. 2018. Feedback-guided anomaly discovery via online optimizatiaon. In *Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining*. 2200–2209.

[140] Shai Shalev-Shwartz et al. 2012. Online learning and online convex optimization. *Found. Trends Mach. Learn.* 4, 2 (2012), 107–194.

[141] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. 2001. Constrained k-means clustering with background knowledge. In *Proceedings of the (ICML'01)*, Vol. 1. 577–584.